AALBORG UNIVERSITY COPENHAGEN

Programme: Medialogy

Semester: 10

Title: Cooperative Play – The Effect(s) of Cooperation in a Digital Game Based Learning Environment

Project Period: 1st February – 2nd June 2017

Semester Theme: Master Thesis

Aalborg University Copenhagen
A.C. Meyers Vænge
2450 København SV, Denmark

Semester Coordinator:

Secretary:

Supervisor(s): Stefania Serafin

Project group no.: N/A

Members:
Ryan Anderson
Christian S. Hansen

Abstract:
Game-based learning (GBL) is a newly-emerging learning paradigm that facilitates the accomplishment of learning objectives and promotes an increasing number of motivated learners. Due to GBL's undisputable advantages, the presence of digital game-based learning (DGBL) in the modern-day classroom is becoming increasingly imminent. Thus, this study investigates two research questions pertinent to DGBL and cooperative learning: 1) does a cooperative DGBL environment provide more effective learning experiences than that of an individual DGBL environment? 2) Is the process of cooperation more effective over a virtual medium than in a common physical space in cooperative DGBL? The experiment was conducted on 24 third grade school children at Lindehøjskolen in Herlev, Denmark. The experiment utilised the scientific control method to answer both research questions. The first research question observes the following dependent variables: motivation, speed, and precision, when altering the learning environment. Research question two examines the following dependent variables: cooperation, speed, and precision, when manipulating the cooperation medium. The results of the experiment conveyed no statistical significance for both research questions: 1) motivation (p value = 0.67), precision (p value = 0.18), speed (p value = 0.43). 2) Cooperation (p value = 0.80), precision (p value = 0.75), speed (p value 0.68). Therefore, these results indicate that a cooperative DGBL environment does not provide more effective learning experiences than that of an individual DGBL environment, and that the process of cooperation is just as effective in a common physical space than over virtual medium.

Copies: 1
Pages: 101
Finished: 1st June 2017

Cooperative Play - The Effect(s) of Cooperation in a Digital Game-Based Learning Environment

INDIVIDUAL GAME-BASED LEARNING VS. COOPERATIVE GAME-BASED LEARNING

Ryan J. Anderson and Christian S. Hansen

Abstract

Game-based learning (GBL) is a newly-emerging learning paradigm that facilitates the accomplishment of learning objectives and promotes an increasing number of motivated learners. Due to GBL's undisputable advantages, the presence of digital game-based learning (DGBL) in the modern-day classroom is becoming increasingly imminent. Thus, this study investigates two research questions pertinent to DGBL and cooperative learning: 1) does a cooperative DGBL environment provide more effective learning experiences than that of an individual DGBL environment? 2) Is the process of cooperation more effective over a virtual medium than in a common physical space in cooperative DGBL? The experiment was conducted on 24 third grade school children at Lindehøjskolen in Herlev, Denmark. The experiment utilised the scientific control method to answer both research questions. The first research question observes the following dependent variables: motivation, speed, and precision, when altering the learning environment. Research question two examines the following dependent variables: cooperation, speed, and precision, when manipulating the cooperation medium. The results of the experiment conveyed no statistical significance for both research questions: 1) motivation (p value = 0.67), precision (p value = 0.18), speed (p value = 0.43). 2) Cooperation (p value = 0.80), precision (p value = 0.75), speed (p value 0.68). Therefore, these results indicate that a cooperative DGBL environment does not provide more effective learning experiences than that of an individual DGBL environment, and that the process of cooperation is just as effective in a common physical space than over virtual medium.

Introduction

With the ubiquitous influence digital technology has on our everyday lives it is intriguing to investigate how in an educational domain, digital solutions can enhance learning experiences. Nowadays, children frequently interact with digital technologies as a form of convenient entertainment, and on occasion for the purpose of education, as part of their learning curriculum or homework. As a result children have learnt how to quickly adapt to these new appealing technologies (Rieber, 2001).

Inevitably, children will learn basic cognitive and social skills at school as part of a mandatory step of human self-development and self-maturation. For a minority of children who attend school there runs a risk that as they progress through the school's curriculum they may struggle with self-development, also negatively impacting their attitude and interest towards learning, which as Rieber (2001) describes it: *"is regimented, homogeneous, and based more on rewards and threats than curiosity and interest;"* therefore, as a result disruptive learning environments may ensue, not only for the apathetic child itself but also influencing the entire classroom. The question then arises; can digital technologies facilitate learning at school, regardless of a child's cognitive ability? By hosting formal educational exercises on digital platforms, and introducing game-based learning (GBL) into the classroom learning environment it may help promote more motivated, intriguing and curiosity-driven learning experiences.

In recent years, research has been performed relating to the validity of incorporating GBL into existing traditional learning frameworks (Sawyer, 2002; Meluso et al., 2012; Kiili, 2006; Hamari et al., 2016; Barzilai & Blau, 2014; Admiraal et al., 2011), conveying positive results.

The focal point of this investigation is as follows: scrutiny of prevailing game design guidelines, learning methodologies and mathematical didactics. How to provoke and sustain learning motivation, the effects of cooperation, and what significance does the cooperation medium have on the process of cooperation in digital game-based learning (DGBL), and of course the development of an instructional game to assess the subsequently noted questions. The instructional game will incorporate novel game characteristics with reconstructed learning content influenced by traditional mathematical exercises located in third grade mathematical school books (to ensure appropriate game difficulty). The purpose of this investigation is not to compare learning experiences in a GBL environment and classroom learning environment, but instead to observe the effects of introducing cooperative learning in a DGBL environment. Therefore, this experiment will initially investigate learning experiences in individual DGBL versus cooperative DGBL, and then further explore whether the manipulation of the cooperative medium in cooperative DGBL has any significance on the success of cooperation (and learning experiences): With these ambitions and research questions in mind, the following problem statement can be formulated:

"How can we design and develop a cooperative DGBL experience? Is a cooperative DGBL experience more effective than an individual DGBL experience, and is the process of cooperation more effective over a virtual medium than in a common physical space?"

In this problem statement, it is important to define what we imply when we say "effective;" in the instance of a DGBL experience an effective learning experience was defined using three dependent variables: motivation (how motivated the individual was to learn) precision (how accurate was the individuals' answers), and speed (how quickly the game is completed). Whereas in the instance of effective cooperation, it was also defined via three dependent variables: cooperation (how cooperative was the individual), precision (how accurate was the individuals' answers), and speed (how quickly the game is completed).

Prior to exploring these research questions, it may be beneficial to investigate a number of distinctive ideologies regarding the definition of the term *game*, after which it will be possible to present a more universal perception and interpretation of the term, *game*. Perhaps the most acclaimed and traditional interpretation of games was described primitively by Caillois (1961) as *"an activity that is voluntary and enjoyable, separate from the real work, uncertain, unproductive in that the activity does not produce any goods of external value, and governed by rules."* Crookall et al. (1987) shared a very similar opinion and describes games as having the attributes of clear rules, goals, and strategies, and in the event of goal achievement the game is terminated, and a winner and loser is determined (and in some cases, neither can be determined, due to a draw); where losing

only has consequences in the game world. Gredler (1992) also possesses a similar notion; he defines a game as an activity that introduces a contest between individuals, operating under regulated conditions with a purpose to achieve an objective. In contrast to Crookall et al. (1987), Sawyer (2002) suggests that both rules and strategies may be ambiguous at game start, as discovery is an important element of a game experience. Garris et al. (2002) argues that there are six fundamental aspects that describe the characteristics of games: fantasy, rules/goals, sensory stimuli, challenge, mystery, and control, and that if these characteristics are implemented and combined with instructional content, then the learning experience can be perceived as being more game-like. Previous research has suggested that games have the potential to be a compelling educational tool which can be used to enhance learning experiences and to effectively learn complex subject matter (Cordova & Lepper, 1996).

GBL is a new emerging ideology in the field of learning, games which incorporate GBL typically have predefined learning objectives, and combine standard instructional learning content with game characteristics in a fantasy game-like setting which promotes increased student interest and learning motivation (Druckman, 1995). As briefly mentioned previously, the goal with GBL is to promote motivated learners, as motivated learners are committed students whom have a passionate and focused attitude towards learning, as they are genuinely interested in the learning task; they enjoy what they are doing, they are diligent, and they are determined. Due to the intense motivational properties and heightened attention state of individuals that a game experience offers, the goal of learning through games is to manipulate this motivational energy in a manner that positively impacts the student's attitudes towards learning. As GBL enhances student motivation, it also advantageously promotes greater attention and thus resulting in greater knowledge retention (Ricci et al., 1996), this is due to GBL offering a more learner-centred model which encourages active learning and to learn by doing, rather than learning by listening.

The phenomenon of motivation has been described as the "driving force" which helps us to achieve our goals (Lawyerment, 2017), and by Wolters (1998) expressed as *"an individual's choice to engage in an activity and the intensity of effort or persistence in that activity."* Deci and Ryan (2000) suggest that motivation is an instinctive human psychological need that can be characterised by three variables: competence, autonomy and relatedness, and describes motivation as a *"need"* to pursue goals. Deci and Ryan (2000) states that curiosity is a crucial component of motivation and is considered a primary element to facilitate personal growth and self-determined learners, and similarly Berlyne (1960) believes that curiosity is a primary factor that drives learning, and is usually evoked for one of two reasons: sensory curiosity or cognitive curiosity. Additionally, Kashdan et al. (2004) has defined curiosity as *"a positive emotional-motivational system associated with the recognition, pursuit, and self-regulation of novel and challenging opportunities."* Although researchers have come to the consensus that motivation exists in two distinct forms, *Intrinsic* and *Extrinsic*; Hence why Wolters (1998) defines motivation as the *"individual's choice to engage in an activity"* and the *"intensity of effort"* exerted in the activity, as these statements define the key elements that make it possible to distinguish between intrinsic and extrinsic motivation. Deci and

Ryan (2000) suggest that intrinsic motivation is driven by self-determined activities which are activities that people execute naturally when an inner-interest for the activity exists. He also elaborates that intrinsic motivation can be hindered by extrinsic rewards and external pressure or optimised by aligned individual skill and task challenge, otherwise called *"flow"* (Csikszentmihalyi, 1975). Whilst Malone (1980) states that to make an activity intrinsically motivating it must support at least one of the following factors: challenge, curiosity, and fantasy. Furthermore, supporting previous definitions of extrinsic motivation, Vallerand and Fortier (1997) have described it as motivation that is stimulated by participating in an activity as a means to an end.

Another term that is pertinent to this investigation that must be defined is *cooperation*. The activity of cooperation has been defined by Johnson et al., (2013) as when a set of actors, as a group, work together to accomplish a commonly-shared goal. The goal to be attained can be relevant for the individual actor in two manners: 1) it can be beneficial to the individual actor. 2) It can be beneficial to the group (Johnson et al., 2013). An individual's effort to indulge in cooperation may depend on multiple social and task related factors defined by individualism-collectivism (Wagner, 1995). By Wagner's description, an individual whom is more individually inclined can be described by individualism; they are characterised by an increased interest in individual gain and considers personal interests to be of greater importance in relation to group interests. Of course, an individually inclined person may still convey gestures of cooperation when personal goals are aligned with group goals. Contrastingly, when a person is more motivated by group interests they can be defined by collectivism; collectivists determine group interests and goals to be of greater importance over their own.

This investigation attempts to seek a more supportive and motivated learning environment by combining two factors: cooperative learning (distinctly different from individual and competitive (Johnson et al., 2013)) and DGBL; thus, ensuring shared responsibility of achieving learning objectives in a game-like environment.

To challenge the proposed problem statement, we developed a mathematical probability game *"Plain Probability."* The instructional game was also used as a tool to generate data that represent the corresponding dependent variables of speed and precision, and further qualitative data was gathered from post-test questionnaires which were used to quantify the phenomenon and activity of motivation and cooperation.

Related Work

Cooperation

Cooperation has been defined as a process where a set of actors work together to accomplish a common goal. Goals in nature may have an appeal to a group of people (a team) or alternatively just a sole individual and the accomplishment of goals typically benefit the actors whom achieve them, if they bear relevance to their interests. The activity of cooperation may exist and be applied in situations of varying time frames, ranging from a team of people participating in a day workshop to a large department of 100+ employees working on a long-term project. These situations are seen in different

implementations of cooperative learning (e.g. peer-assisted learning), complimenting each other (Johnson et al., 2013). By its very nature, cooperative learning can be described as inversely relative to competitive learning where actors work against each other to achieve a common goal.

Effects of cooperation presented by Deutsch (2001) are also relevant in cooperative learning and can be defined by the following: communication, friendliness and helpfulness, coordination of effort, division of labour, group satisfaction with agreed ideas, recognition and respect of the other's needs, willingness to enhance others (knowledge and skills), and defining conflicting interests as a mutual problem to be solved by collaborative effort. In summary, successful cooperation results in the elicitation of more cooperative effects, increased team cohesion, and a unified work environment where team members are able to share their beliefs and attitudes. Whereas in an environment where individuals work only for their own benefit completely disregarding any activity that promotes group cohesion (absence of cooperation), the following effects can be observed: impaired communication (due to conflicting interests), distrust, obstructiveness and negative attitude, rejection and disagreement of ideas, overlapping and duplication of effort, and viewing weaker team members as a liability to the group.

An individual's stance regarding individual and group interests can be determined by analysing individualism-collectivism (Wagner, 1995). Furthermore, Wagner suggests that people can either consciously or unconsciously act on individual interests, defining the two distinct situations as free riding or social loafing, respectively.

The theory of cooperation derives from social interdependence theory (Johnson et al., 2013), which suggests goal structures determine how individuals act and as a result create outcomes. Therefore, social interdependence has been categorised into three specific modes: positive interdependence, negative interdependence, and no interdependence, which distinguish between individual actions because of differing goal interdependences.

Social Interdependence

Social interdependence originates from the idea that groups are dynamic wholes, where each member of the group possesses a unique level of interdependence. Previous research cited by Johnson et al. (2013) elaborated on social interdependence theory and describes the "dynamic whole" as the essence of a group constructed of a collection of interdependences relevant to the common goals of each team member. Due to the dynamic nature of social interdependence, the alteration of the state of a group member or sub-group may also influence the state of others. Moreover, the intrinsic state of tension between group members is advantageous as it acts as a motivator towards the progress of shared goal(s) (Johnson et al., 2013). The original theory of social interdependence has been further expanded upon by Morton Deutsch, in which he terms *"cooperation and competition"* (Deutsch, 2001, Johnson et al., 2013). Deutsch specifies three types to social interdependence (see figure 1):

- Positive interdependence (cooperation)

- Negative interdependence (competition)
- No interdependence (individualistic efforts)

Positive interdependence can be described when the interdependence between group members is based on unity and that they believe that shared goals are only obtainable as a result of group efforts. Deutsch (2001) refers to positive interdependence as positive goal interdependence; positive goal interdependence exists when individual goals are linked so that there is correlation between the probability of fulfilling one person's goal and another person's goal. Therefore, a positive correlation exists between both individuals regarding the same goal. The effects of positive interdependence can be described as the development of a positive relationship between group members, the need to share personal resources to overcome group problems, and being rewarded as a result of joint efforts (Deutsch, 2001). Another effect of positive interdependence is the elicitation of *promotive interaction* (Johnson et al., 2013); promotive interaction endorses encouragement between group members as well as facilitating group members' efforts to learn and develop.

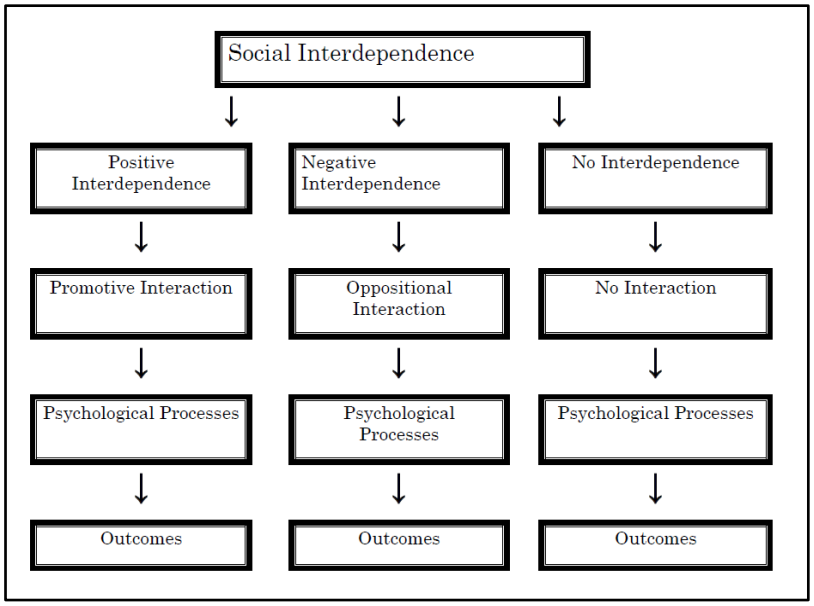


Figure 1 - Overview of Social Interdependence theory (Johnson et al., 2013).

Negative interdependence is elicited in competitive environments or controversial group environments, where individuals are opposed to each other due to the inverse correlation between their goals. Therefore, there are two situations where negative interdependence exists: 1) Individuals possessing inversely correlated links to the same group goal, thus hindering each other's efforts. 2) Individuals possessing inversely correlated links to the same goal, where only one individual may achieve the full reward (competitive). Deutsch's (2001) definition of negative goal interdependence similarly explains the negative link between individual goals, where the probability of one person's goal being achieved is negatively correlated to another persons' goal achievement, and thus one can only achieve their goal when the other fails. As Johnson et al.'s social interdependence structure conveys, negative interdependence results in *oppositional interaction* (Johnson et al., 2013), where one attempts to obstruct or

discourage another individuals effort towards their goal achievement. Therefore, negative interdependence results in the following effects: a genuine dislike for another person, and evocation of a competitive attitude. No interdependence exists in an individual setting, where there is no correlation between the probability of achieving one person's goal and another persons' goal, due to the non-existence of a second individual. No interdependence exists when an individual works alone to achieve a goal, absent of help (cooperation) or obtrusion (competition) with their goal. Social interdependence follows the rule that the form of interactions provoked by individuals is dependable on the interdependence elicited. Therefore, positive interdependence exists as a result of promotive interaction, negative interdependence exists as a result of oppositional interaction, and no interdependence exists when there is no interaction (Johnson et al., 2013), as seen in figure 1. Interdependence has been presented on two extremes; therefore, when multiple goals are presented to an individual they may possess a collection of positive and negative interdependences relative to each goal.

Deutsch (2001) presents two forms of action that one may perform in order to affect the probability of a goal for themselves and other people: *effective actions* and *bungling actions*. Effective (or promotive) actions are progressive, improving the chances of success and goal achievement for the actor, whereas bungling (or obstructive) actions are the opposite, worsening the actor's chances. An effective action can become a bungling action towards another if the two participants have a negative interdependence; this is because the person helping himself (the effective action) hinders the other (who sees it as a bungling action). Similarly, the inverse can occur; having a person taking a bungling action towards them self is an effective action by the other participant. The two basic actions can in conjunction with interdependence affect the following three basic psychological processes, which can in turn be specified as positive or negative dependent on the interdependencies (Deutsch, 2001; Johnson et al., 2013):

- Substitutability
- Attitudes
- Inducibility

These processes are important to understand the social and psychological processes that create major effects of cooperation and competition (Deutsch, 2001). Substitutability is described as *"How a person's actions can satisfy another person's intentions"* (Deutsch, 2001), the individual's active reaction towards other's efforts, and focuses on how a person's action can possibly promote or obstruct another's intentions. Positive substitutability is when there is acceptance and encouragement towards another's activities and their effects, while negative substitutability rejects the efforts and activities done by others. Attitudes are defined as an individual's response, and how the individual invests his energy towards himself and the environment. One can tend towards a positive attitude, having a state of mind that supports cooperative solutions, help, support and other similar tendencies. Or one can have a negative attitude, being of a competitive mind set, where distrust and opposition is more likely. Lastly, inducibility relates to the ease and readiness towards external influences, as well as the

acceptance of influencing others. Positive inducibility is when the individual is positive towards influence and accepts external influences allowing others to do what they want. Negative inducibility on the other hand are not open to influence, rejecting other's influence and their wants. The process of inducibility complements substitutability as an individual will be willing to help other helpful participants or reject helping another during an obstructive action. As such, participants in a cooperative situation will substitute for each other and keep a positive attitude and emotions. They are open for influence and change from others who aids their attainment of their goal. Contrastingly in a competitive situation, individuals work more independently; not substituting for others, not caring for others, and rejecting another's attempt to influence them. None of these presented psychological processes occur during individualistic situations where there is no interdependence (Johnson et al., 2013). Lastly, it is worth noting that as these presentations describe individuals, it is possible to have similar effects between groups. For example, two groups can internally be cooperating, but participate in a competitive situation externally between each other.

Individualism-Collectivism

As presented earlier, Wagner (1995) has determined an analytical dimension, the individualism-collectivism spectrum (see figure 2); investigating the relationship between personal interests and shared goals. On one end, individualism relates to personal interests and the greater importance of these interests over other cooperative or group interests as well as the missing need for participating in cooperative situations, rather accomplishing one's goals individually. In the other end, collectivism and the collectivists drawn to this side of the spectrum find the greatest importance in the group interests. Here personal interests are secondary, and one wants to work together to accomplish the goals set for the group. Wagner (1995) presents a definition of a *"self"* as a term for how individualists and collectivists look at the important entity. Individualists regard *self* as a singular element, one person. Selfishness for *self* implies attention to the personal interests and a pursuit of these, while disregarding group interests. Collectivists on the other hand regard *self* as the whole of the group, not the individual participants. Here selfishness prioritizes the accomplishments of the group goals and interest while keeping inattention towards the personal goals.

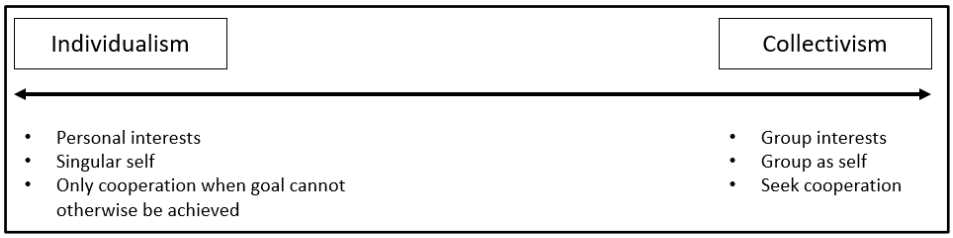


Figure 2 - Spectrum of Individualism-Collectivism.

As these representations of individualism and collectivism are both to the extremes of the spectrum, the actors present can have mixed amounts of individualistic and collectivistic interests and variations in these amounts can determine the want and/or need for cooperative actions in group situation. Individualists might seek out

cooperation to achieve personal interests; being an extremist on the spectrum does not force rejection of cooperation and group work.

Risks in a Cooperative Setting

In a cooperative setting, there is a risk of individuals not carrying their own weight, letting the rest of the group handle the work needed to attain the group's goal. This risk can present itself in two forms: *free riding* and *social loafing* (Wagner, 1995). These forms share many aspects, with the difference in how they occur. Free riding is an abuse of and an impairing effect on cooperative work. An individual acting on free riding chooses to avoid the cooperative elements in the group where the individual is in, letting the other members do all the work, yet still pursuing the rewards presented from the work done, acquiring rewards based on others' work. This free riding can result in reduced individual performance and effort and ruining cooperation, decreasing group performance and well-being. Social loafing shares similar functions to free riding where an individual does not carry their own weight whilst still possessing the expectations to reap the rewards of a team effort. The difference is then in the occurrence - how the individual arrives at the conclusion that free riding or social loafing is a "good idea". Here social loafing is an unconscious choice, with the individual possibly not knowing their position. Free riding is then when the individual consciously decides to act accordingly out of rational thought. Wagner (1995) presents a set of variables that can affect the probability of these risks, namely:

- Group size
- Identifiability
- Shared responsibility

Variation in these variables may increase the risk of free riding and social loafing if the group allows for an individual to disguise their lack of performance. Group size can increase the risk when these are of a larger scale. When more people participate in a task, an individual can dedicate less effort to that task and as such, this larger group has helped towards producing free riding or social loafing. Smaller group sizes on the other hand forces the individual to participate if he wants to achieve his goals and getting the rewards. Identifiability is a variable that defines the ease at which an individual and his tasks can be identified. The higher the identifiability, the more transparent an individual is, making it harder for them to hide from others. Low identifiability establishes individual anonymity, so that they cannot be targeted as a reason for worsened group performance. Lastly, shared responsibility responds to variation in personal responsibility. As such, a high shared responsibility reduces the personal responsibilities towards a task, as more members share the responsibility to finish the task, allowing individuals to reduce their performance. With a lowered personal responsibility, the individual is more likely to feel the group has no need for his work. Low shared responsibility forces members to all perform better, as less members work on a task, the individuals' performances are more visible and they feel the group needs them to participate to reach their goal and succeed.

Implementation of Cooperative Learning

Implementing cooperative learning in an existing setting is not just done by selecting a set of people and placing them in a group. To possibly create cooperation, one must have the formed group to interact, as it is through interaction in combination with a set of essential conditions that cooperation can occur (Johnson et al., 2013). The conditions are: positive interdependence, promotive interaction, individual accountability, use of social skills and group processing.

As presented earlier positive interdependence occurs when group members share goals and these goals are reachable through a group effort. As such, this interdependence promotes cooperation for the sake of goal accomplishment. Individual accountability can exist when individuals' performance is observable and accountable. This condition compliments Wagner's (1995) definitions of identifiability and shared responsibilities, as a positive outcome of cooperation requires a joined effort by all. Ensuring every member to be accountable leads to the individual's' learning themselves as well as helping others learn. Individual accountability can be structured through tests of group members, peer-reviews and observations (Johnson et al., 2013). As a result of positive interdependence, the group members should enact promotive interactions, further strengthening themselves and the group. These interactions help development of one's cognitive processes through explanation and assistance, as well as social skills developed by the social interaction naturally occurring during cooperation. The use of social skills contributes to the cooperative effort and success by promoting decision making, trust, leadership, conflict management and communication. Using and developing these social skills leads to a more fruitful cooperation between members. Finally, group processing is the ability to assess and consider other group members' process. This element is used by the group to define helpful as well as disruptive actions as to ensure an as smooth learning process as possible. Three general implementations of cooperative learning can be discerned:

- Formal Cooperative Learning
- Informal Cooperative Learning
- Cooperative Base Groups

All subsequent groups characterised by varying timespans aspire to use cooperation and its sub-criteria to improve on self and group learning (Johnson et al., 2013). Informal cooperative learning is the shortest implementation. Here a group work together to achieve a learning goal over a time span of minutes to a class period. The group can be formed to tackle a set of challenges presented focusing their attention to the given material ensuring learning through repetition, discussion and summary. This option for learning can well be used to present closure of a topic. Formal cooperative learning spans between a class period and up to several weeks. This learning method focuses more on assignments and cooperation to complete larger tasks. The longest learning period belongs to the cooperative base groups. Here a group is formed and works together over long term, which requires stable membership to accommodate a diverse set of

cooperative tasks. It is noteworthy that all three methods of cooperative learning support each other and can be used in conjunction.

According to Johnson et al. (2013), to ensure the best performance of the learning methods, the instructional unit (teacher or digital technology) must pre-determine objectives and group structure, deciding on group size, roles, the environment and materials. The unit must further explain tasks, preparing the groups with the needed information of concepts and strategies to complete the tasks, as well as specifying the concepts of positive interdependence and individual accountability and giving criteria for success. During work, the instructional unit should observe and be prepared to provide aid as to improve interpersonal and group skills. Finally, after work, the unit should help process what was learned and evaluating the group to determine how improvements can be made. This evaluation can also include intergroup discussions on effectivity and cooperation.

Cooperative learning is the basis for other forms of active learning: problem-based learning, team-based learning, collaborative learning and peer-assisted learning. Problem-based learning focuses more on the learning process than the results; using small groups having them work together to master procedures and information relevant for the given task. Using a basis of competition or individualism lends to worse results than cooperation. Team-based learning focuses on using small learning teams to improve quality. The teams make use of members with diverse skill sets in a more permanent setting. Members are individually accountable for assignments relevant for the task and must partake in team efforts. Collaborative learning focuses on natural learning with intuitive responses towards the needed efforts. Learning happens interpersonally, by use of dialog and discussion. Lastly, peer-assisted learning focuses on learning through aiding others, here the teaching member consolidates their knowledge by reviewing another's work; the recipient must be open to influence to allow for cooperation.

Motivation

Motivation exists in two states, intrinsic and extrinsic motivation. Therefore, the phenomenon of motivation can be described as a dichotomous human psychological structure, and as stated previously is simply an internal driving force that supports an individual to achieve a specific goal. Motivation is a crucial psychological structure that must be aroused within learners to stimulate successful learning, and encourage inquisitive, focused and enthusiastic students. Malone (1980) and Deci and Ryan (2000) have arrived at the consensus that an activity that is said to be extrinsically motivating is an activity that is motivated by external rewards such as money, encouragement or social status; and in contrast, an activity that is said to be intrinsically motivating is an activity with no discernible reward. In some cases, this distinction between intrinsic and extrinsic motivation leaves the term vulnerable to contradicting interpretations and ambiguity. Instead of defining intrinsic motivation as a process that is not influenced by external reward, it may better be defined as self-determination and a need for competency (Deci & Ryan, 2000) or when an individual experiences *"flow"* (Csikszentmihalyi, 1975).

Self-Determination Theory

Deci and Ryan (2000) state that self-determination theory (SDT) is a psychological theory that attempts to understand human motivation in relation to people's' motivation to pursue goals. SDT can be characterised by respecting three basic human psychological needs:

- *Competence*, the need for challenge.
- *Autonomy*, the need for personal freedom and decision making.
- *Relatedness*, the need to interact with others.

Deci and Ryan (2000) then further elaborate that an activity that is deemed to be interesting by an individual is driven by intrinsic motivation as the activity complements the person's inner interests. In comparison, an activity that is not necessarily of interest to someone but is believed to be of importance is more inclined to be motivated by extrinsic motivation, as the person feels that the task must be completed. Deci and Ryan (2000) also suggests that when extrinsic rewards are introduced (either tangible, or not) to a previously intrinsically motivating activity, the activity then becomes extrinsically motivated as the activity is being controlled by the reward rather than the person themselves, resulting in a lack of individual autonomy. Therefore, as intrinsic motivation concerns active engagement with tasks that people find interesting, it is logical to consider that when someone experiences *"flow"* (an optimal level of challenge) (Csikszentmihalyi, 1975) they are also experiencing intrinsic motivation. Previous research cited by Deci and Ryan (2000) also demonstrated that activities that provided positive feedback enhanced intrinsic motivation when being compared to activities that provided no feedback at all. While activities that provided negative feedback decreased intrinsic motivation relative to no feedback. Additional studies by Fisher (1978) also determined that positive feedback only enhanced intrinsic motivation when the individual felt responsible for their own performance or when the individual was free to make their own decisions. Thus, these results imply that positive or negative feedback rendered to an individual has an impact on the person's perceived level of competence, also impacting the level of intrinsic motivation for the person's current activity. Although competence and autonomy has been defined as two crucial psychological needs of intrinsic motivation, research has shown that relatedness also is a minor factor when maintaining intrinsic motivation. This factor has also proven to be more significant when associating intrinsic motivation with children, as a study by Ryan et al. (R. Ryan, Stiller, & Lynch, 1994) depicted that children whom have experienced friendly and caring teachers showed a greater level of intrinsic motivation for their school work. Deci and Ryan (2000) then summarise that for any type of motivation to take place, competence is a necessary psychological need, and to translate that motivation into intrinsic motivation the individual must be able to express their need for freedom and personal decision making without restriction or control, and that the level of intrinsic motivation can be enhanced by relatedness, and positive feedback received from others.

Concept of Flow

The concept of flow was conceived in 1975 by Csikszentmihalyi (1975) and has been refined over the last four decades. Flow is a state that can be aroused within an individual when participating in an enjoyable activity, and thus due to the intrinsically motivating nature of the activity, the individual becomes deeply absorbed and focused on their play and performance. Therefore, the *"flow"* experience can be described as a state of intense concentration where an individual is so focused on an activity that they are less likely to be distracted from the activity, in some cases the individual may experience temporal incongruency, and as a result time may become distorted. Due to the enjoyable nature of the activity, the individual is motivated intrinsically to complete it whilst totally disregarding whether it is a difficult or dangerous task. In an educational context, the state of absorption and high concentration can be utilised to promote optimal learning experiences. Therefore, someone who is said to be *"in flow"* (experiencing flow) can be observed as possessing the following characteristics (Csikszentmihalyi, 1996):

- Intense and focused concentration on the current activity.
- Not easily distracted from the current activity.
- Action and awareness is unified.
- Loss of self-awareness and physical existence.
- Having no fear of failure.
- Loss of temporal perception (time elapses faster than normal).
- The current activity is intrinsically rewarding; thus, the end goal is a justification to proceed with the activity.

Furthermore, it has been clarified that the following rules must be respected in order to elicit the optimal experience of flow (Csikszentmihalyi, 1996):

- The challenge that the individual encounters must align with their current possessed skill set (neither too easy nor too difficult); the individual must sense that the challenge is of an appropriate difficulty that matches their capacity.
- The goals of the challenge are clear and feedback regarding the progress of the current goal is immediate.

Bandura and Schunk (1981) also support this notion and stated that short-term goals were more significant than long-term goals when sustaining motivation and curiosity for an activity.

The balance between challenge and individual skill has been described as a fragile matter by Admiraal et al. (2011), and is perhaps the most significant factor to consider when inducing flow. As figure 3 illustrates, an individual can experience anxiety when the level of challenge in the activity is greater than their level of skill (ability), or in other words the activity is too difficult. Whereas in contrast, if an activity is deemed to be too easy the individual may become bored with the current activity, since their level of skill is greater than the challenge presented in the activity. Lastly, an individual may witness

apathy, due to the lack of their skill and challenge of the current activity, and is simply the antithetical state of flow.

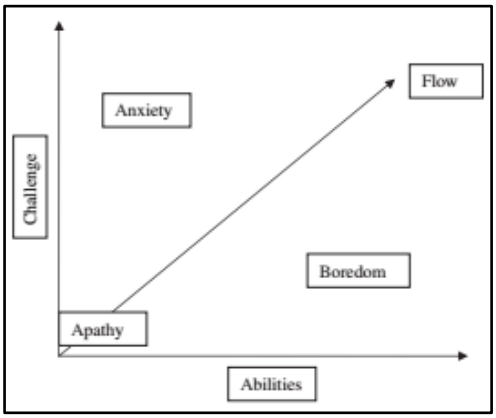


Figure 3 - The flow model (Malone, 1980).

In summary, the relationship and balance between challenge and ability (the concept of flow) has been used by sports coaches, teachers and designers for decades and now more recently by video game developers and the field of eLearning, in order to promote optimal performances.

Flow in Game Based Learning

Killi (2006) has conceptualised a flow framework for GBL from experiential learning theory, game design, and flow theory. The model emphasises the importance of implementing factors of flow theory in GBL, and highlights the need for immediate feedback, and distinct goals that correlate to the learner's perceived skill level. According to the model (see figure 4), the likelihood of experiencing flow relies upon the interaction between the task, person, and the artefact (the tool or toy that hosts the task).

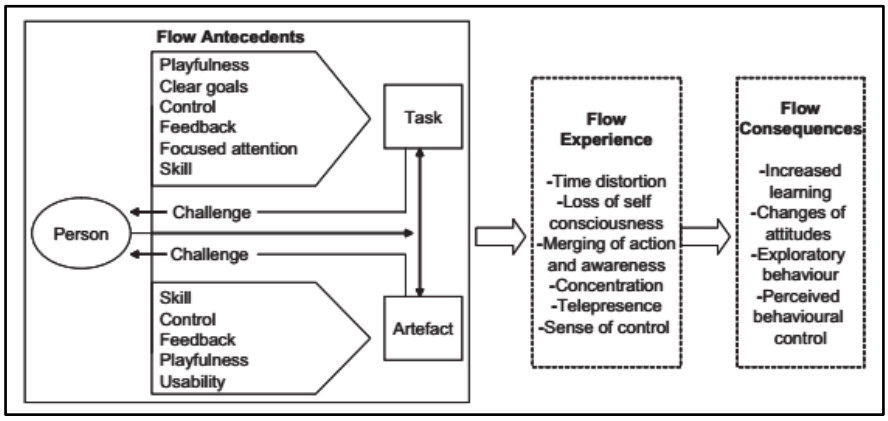


Figure 4 - A framework of flow in virtual environments (Kiili, 2006).

Killi (2006) defines that a task which supports flow can be characterised by supporting the following traits: *playfulness, clear goals, control, feedback, focused attention*, *skill*. Additionally, an artefact that is considered to facilitate flow has the following attributes: *skill, control, feedback, playfulness*, and *usability*. Similarly to what was explained by Csikszentmihalyi (1996), Killi (2006) states that when one experiences flow in a virtual

environment, they may experience the following phenomena: time distortion, loss of self-consciousness, merging of action and awareness, concentration, telepresence, and sense of control. Lastly, the framework illustrates the consequences of flow, or in other words the outcomes of experiencing flow: increased learning, changes of attitudes, exploratory behaviour, and perceived behavioural control.

Curiosity

Curiosity can be interpreted as an essential emotional component that must be evoked within an individual if motivation is to take place. According to Malone's (1980) perception of curiosity (adapted from diverse and specific curiosity (Berlyne, 1960)), curiosity is absolutely paramount in order to maintain an intrinsically motivating activity. Although, for an activity to remain curious to us, it must sustain an optimal level of informational complexity, in other words the activity must neither be too simple nor too complex but just right with respect to the student's existing knowledge. Optimally, the activity should provide the learner with an idea of what to expect, but whether those expectations are met, should remain uncertain. Berlyne (1965) suggests that a prominent factor that leads to curiosity is a factor that he calls, *conceptual conflict*. Conceptual conflict is a phenomenon that is induced when an individual hears of an idea that contradicts with what was previously known. For example, if someone who hears about a squirrel that can fly (a flying squirrel), and previously they believed that squirrels could not fly, conceptual conflict in addition to curiosity is induced. Similarly, Malone (1980) believes that curiosity can be divided into two types of curiosity: *cognitive curiosity*, when the individual believes that there is incongruent or absent information in their knowledge structure, and *Sensory curiosity*, when the current experience is peculiar to the senses.

Cognitive Curiosity

Cognitive curiosity is the phenomenon of attracting attention by convincing an individual that their existing knowledge is incomplete, inconsistent, or unparsimonious, and as a result they wish to improve their knowledge. This then leads to three coherent cognitive theories: *completeness*, *consistency*, and *parsimony*.

Completeness

Knowledge completeness is a strong cognitive curiosity to locate all necessary information to complete an internal knowledge structure.

Consistency

Similar to Berlyne's (1965) *conceptual conflict* theory, knowledge consistency is curiosity that is evoked when inconsistencies of an individual's knowledge is made apparent to them, thus intriguing them to discover further inconsistencies in their knowledge structure. For example, students may be told that mammals do not lay eggs, however the Australian duck-billed platypus is a mammal that does lay eggs (Dasgupta, 1998), therefore conveying an inconsistency of the student's current knowledge.

Parsimony

Knowledge parsimony evokes curiosity by teasing an individual with small pieces of information, one at a time. Therefore, knowledge parsimony is more commonly induced

when inductive instruction (bottom-up approach) is used by a learner instructor, as inductive instruction shows several examples of how to use a generic rule before presenting and explaining the new general rule (Bilash, 2009).

Sensory Curiosity

Sensory curiosity is the phenomenon of attracting attention via the manipulation of sensory stimuli such as: light, sound, vibration, or even smell.

Instructional educational games exist in virtual environments; therefore, effective environmental stimuli can be fabricated with ease. This presents several diverse possibilities, and thus enables eLearning developers to manipulate environmental stimuli in virtual environments to promote curious learners.

Learning Engagement

Instructional games, gamification and GBL are distinctively different from games with an entertainment-centred approach, whilst they are enjoyable, they are typically designed with educational purposes in mind, rather than entertainment. Instructional games incorporate the advantageous attributes of games by combining concentration of challenging activities with enjoyment that is experienced when engaged in a game experience.

OA3 Framework

Schoenau-Fog (2011) investigates the phenomenon of *"continuation desire"*, which he describes as the level of motivation one possesses in relation to how much they wish to continue the game experience. Moreover, Schoenau-Fog elaborates on player motives, and their reasoning for initiating and sustaining engagement in a game-related experience. He further describes the *"player engagement process"* as a distinct four stage cyclical process, thus formulating the objectives, activities, accomplishments, and affect framework (OA3 framework).

Objectives

Objectives are described as a motive or reasoning for players to engage in the game experience in the first place. Furthermore, Schoenau-Fog (2011) delineates two forms of objectives from the objective component of his OA3 framework: *Intrinsic objectives*, objectives that are based from game elements, and are player-defined, e.g. a player that wishes to achieve a certain player level, for the reason of self-gratification. *Extrinsic objectives*, objectives motivated by rewards that explicitly exist in the game itself, e.g. collecting a specific number of items to complete a quest.

Activities

Activities concern the actions a player executes to overcome an objective, Schoenau-Fog (2011) explains an array of nine possible actions players may choose to take: *Solving*; a strategic or tactical activity that challenges one's cognitive abilities, *Sensing*; an activity that stimulates the audio-visual modality by game graphics and audio, *Interfacing*; the activity of enjoying state of the art or unusual game controls or peripherals, *Exploration;* the desire for player autonomy, an activity of game world exploration, *Experimentation*; an activity in which the player indulges in the experimentation of game mechanics and

customisations to discover efficient methods of gameplay, *Creating*; the activity of constructing objects or authoring artwork within the game, with an ambition to continually improve their work, *Destruction*; the activity of destroying objects within the game, typically made more enjoyable when equipped with divine powers, and non-existent consequences, *Socialising*; the activity of socialising with other people for the cooperative purpose of teamwork, and overcoming a common group challenge via the collective formation of game tactics, *Experiencing*; the activity of progression (and in some cases player empowerment) to discover new game experiences (narrative and character development).

Accomplishments

The component of accomplishment is evoked when a player successfully completes an objective. As described by Schoenau-Fog, an accomplishment can take on the form of three different modes: *Achievement*; to unlock new items, acquire access to new game zones, or defeat a difficult boss, *Progression*; to develop the game's narrative and experience narrative milestones, develop a more powerful avatar, or develop personal skill level to heighten the threshold of one's individual competitive skill, *Completion*; to collect every collectable, to acquire all items, to defeat every boss, or the accomplishment of completing an entire game. However, if one fails to complete an objective (accomplishment) then they will bypass the accomplishments component, and be diverted straight to the affect component, (as conveyed in figure 5).

Affect

Affects are emotional states and experiences evoked within an individual during the progress or completion of an accomplishment, Schoenau-Fog (2011) classifies affect as three distinct emotional states: *Positive*, a result of player engagement leading to the evocation of one or more of the following positive emotional states: curiosity, enthusiasm, relief, satisfaction, success. *Negative*, in contrast, a result of player disengagement resulting in the evocation of one or more of the following negative emotional states: frustration, apathy, fatigue, unengaging (due to simplicity). *Absorption*, a heightened state of attention and concentration, when one feels present within the game experience or becomes emotionally attached to characters within the game.

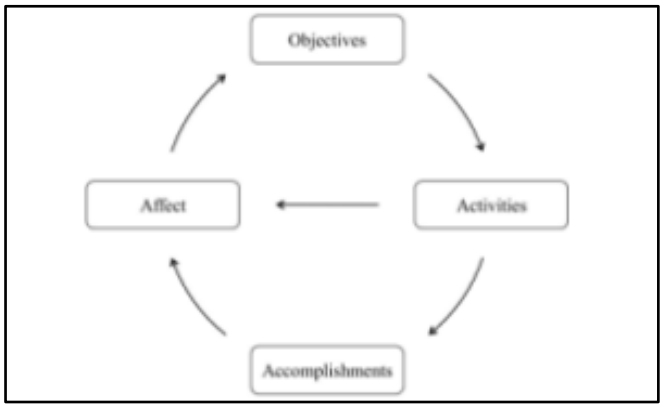


Figure 5 - The OA3 framework conveying the relatedness of the four components of "continuation desire" (Schoenau-Fog, 2011).

Categorising Immersion

Gilmore et al. (1999) suggest that four domains of virtual experiences can be expressed by two common dimensions: *participation*, and *connection*. The axis of participation is represented on a scale from passive to active, while connection is characterised by absorption and immersion (see figure 6), in this context absorption is described as a means to direct the attention of an individual towards the experience. Whilst on the other end of the spectrum, immersion is the hallucinatory experience of when an individual perceives that they are either physically or virtually part of the experience.

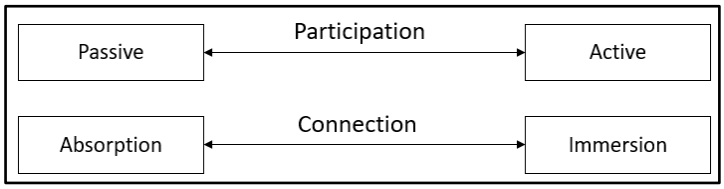


Figure 6 - Two dimensions of virtual experiences (Gilmore et al., 1999).

By considering these dimensions, it is possible to structure four different domains of virtual experiences:

Entertainment

Passive Participation & Absorption

An entertaining experience such as watching television can be categorised by passive participation and absorption, as typically this form of experience requires no interaction from the individual, and only requires their direct attention.

Educational

Active Participation & Absorption

An educational experience can be classified by active participation and absorption, as typically an instructional game requires the individual's direct attention and interactivity.

Aesthetic

Passive Participation & Immersion

The dimensions of an aesthetic experience can be specified by passive participation and immersion, as a heightened appreciation for game aesthetics may result in a sense of presence in the virtual environment, but does not necessarily require player interactivity

Escapist

Active Participation & Immersion

An escapist experience can be represented by active participation and immersion, as an escapist experience requires player interactivity in order to provoke immersion and a sense of presence in the virtual environment. Although concluding on Gilmore et al.'s (1999) thoughts on virtual experience domains, it is disadvantageous to stereotype and categorize virtual experiences as having specific attributes of participation and

connection. An example of such; not all aesthetic experiences can be classified as an immersive experience that do not require player interactivity, in fact in some cases player interactivity may be required in order to fully appreciate an aesthetic experience (event triggered audio feedback, and particle systems as a result of the player interactivity).

Gameplay Experience Model

Ermi and Mäyrä (2005) attempt to make sense of the term *"immersion"* by taking a closer look at the complicated nature of a gameplay experience. As a result of a number of self-evaluation questionnaires filled out by test participants, Ermi and Mäyrä founded three fundamental components of immersion: *sensory*, *challenge-based* and *imaginative* immersion, forming the SCI-model. The consensus of their research was that children were fonder of high quality audio-visual stimuli, and subconsciously thought that well-functioning and well-placed cameras also contributed to an exceptional sensory experience.

Sensory

Ermi and Mäyrä (2005) classified sensory immersion as the ability to be *"immersed"* in the audio-visual aspects of the gameplay experience, including graphics, art styles and audio systems. Furthermore, Ermi and Mäyrä state that the most fundamental component of an immersive gameplay experience was sensory immersion, as even individuals who were unfamiliar with computer games could distinguish the difference between an average sensory experience and an impressive, three-dimensional, well-lighted, stereophonic experience.

Challenge-based

Challenge-based immersion was described as the capability of being immersed by game interactivity mechanics, where Ermi and Mäyrä distinguish two distinct domains of challenge based immersion: *sensomotor challenge*; the sense of challenge derived from challenging game controls and quick reactions, and c*ognitive challenge*; a sense of challenge from problem solving activities.

Imaginative

Imaginative immersion was expressed as one's capacity to become immersed by a game's: characters, narrative discourse, fantasy-like setting, character development or personalities.

Phases of Player Engagement

Brown and Cairns (2004) discuss the ambiguous concept of player immersion within computer games, and define the utmost state of player immersion as *"total immersion"*. Given the nature of most instructional games (embryonic narratives, and low degree of player autonomy, to state a few) it is unlikely that students will experience total immersion during gameplay. However, what is valuable regarding Brown and Cairns' grounded research is the structure and flow of their linear three stage model of player engagement (see figure 7). Previous research also relevant to player engagement by Hamari et al. (2016) did suggest that an elevated state of concentration, interest, and

enjoyment was conducive to learning, although the intensely focused, almost hallucinatory state of immersion did not necessarily convey any positive effect in regards to learning, perhaps due to the human sedative state when experiencing immersion. Furthermore, Brown and Cairns (2004) explain that the transition between states of engagement are controlled by virtual barriers, such as human activity or game qualities, and similarly Hamari et al. (2016) explain that engagement and immersion in GBL depend on the levels of concentration, interest, and enjoyment of the individual, aroused from the level of challenge present in the game and the individual's' perceived skill. Ultimately Brown and Cairns' (2004) interpretation of the three stages of the player engagement model resulted in the following states: *engagement*, *engrossment*, and *total immersion*.

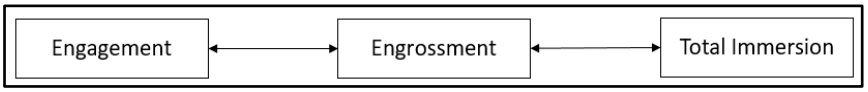


Figure 7 – A linear three stage model of player engagement (Brown & Cairns, 2004).

Engagement

The first stage of the linear three stage model is the state of engagement and is described as the lowest degree of game involvement. Brown and Cairns (2004) further describe that it is only possible for an individual to transit to the state of engagement when the player feels the need or temptation to invest time, effort and attention into the game. Therefore, an engaged individual is someone that wants to continue with the gameplay experience but is not yet attached to the experience on an emotional level. The subsequent statements arise the same predicament that Malone (1980) described in his heuristic game design guidelines, regarding the fact that if an individual dislikes the style of the game, then they simply will not even attempt to engage with it.

Engrossment

Proceeding from engagement is the stage of player engrossment. Engrossment is described as the stage when players attain an emotional attitude towards the game. During engrossment, player emotions can be manipulated by the game whilst simultaneously their awareness of self-existence and their surroundings are also lessened. One participant of the study described engrossment as *"A Zen-like state where your hands just seem to know what to do, and your mind just carries on with the story."*

Total Immersion

Total immersion is expressed as the state of an individual when they feel present in the game environment, and as such are detached from the real world. At this threshold players' emotional states can only be manipulated by experiences in the game. Additionally, individuals may feel emotionally attached to game characters, and thus may be emotionally sensitive towards them. The level of immersion an individual experienced seemed to correlate with the number of game elements that the player found satisfying: visual, auditory and mental.

Engagement and Immersion in Perceived Learning

Hamari et al. (2016) tested *Quantum Spectre*, a physics themed puzzle-style game on 135 high school students as part a fifteen-week engineering course. Hamari et al. (2016) performed structural equation modelling (SEM) and path analysis to estimate any significant causal links between their five variables of interest: *challenge*, *skill*, *engagement*, *immersion*, and *perceived learning*, see figure 8.

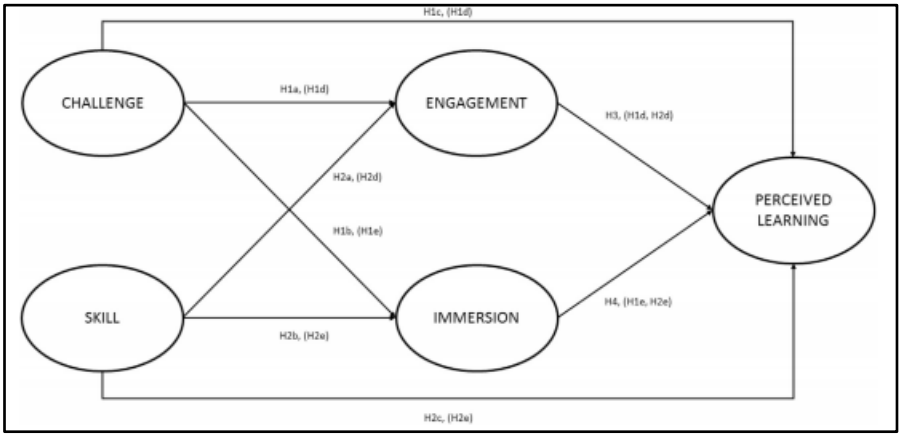


Figure 8 - The structure of Hamari et al.'s path model analysis (Hamari et al., 2016).

The results of Hamari et al.'s (2016) SEM and path analysis supported the following statements:

1. The level of challenge in the game has a positive direct effect on:
    a. Engagement.
    b. Immersion.
    c. Perceived learning.
2. The level of perceived individual skill has a positive direct effect on:
    a. Engagement.
    b. Immersion.
3. Engagement is positively associated with increased perceived learning.
4. The effect of challenge on perceived learning is directly affected by engagement.

From the results, it can be concluded that optimising the level of challenge in an instructional game to support the skill level of a person will result in increased user engagement, immersion, and perceived learning. A person possessing a greater sense of self-efficacy contributes to a higher level of user engagement and immersion; although not perceived learning. Moreover, user engagement positively correlated to an increase in perceived learning, whereas in contrast immersion did not correlate to an increase in perceived learning.

Game Based Learning

As presented earlier, GBL is an ideology making use of games in a learning environment. It makes use of standard instructional learning content while merging it with game characteristics in a game-like setting promoting interest and learning motivation (Druckman, 1995). The reasoning for using GBL is the idea that users can become more engaged and motivated by introducing these game elements, while the GBL solution is still considering the learning goals and the need to reach them. The improvement in engagement leads to better knowledge retention as the game positively impacts attention towards the learning goals (Ricci et al., 1996; Nussbaum, 2010).

GBL is not to be mistaken for gamification, another expanding learning method with its origins from combining games and other standard practices. Whereas GBL makes use of actual game implementations, gamification rather integrates game elements into previously defined practices (Kapp, 2014). As such both GBL and gamification can include the same elements, e.g. score systems. When comparing games with GBL there are some similarities but also several differences that distinguish the two contexts. The purpose of each is completely different; games are built with entertainment as priority whereas in contrast GBL games are built to facilitate learning. However, both contexts share the similarity of possessing game characteristics and objectives. A comparison between games, gamification and GBL can be seen in appendix A1.

Motivation and Engagement

Motivation is vital to ensure a successful GBL experience, as motivation is the primary factor of engaging learning experiences. Motivation can be introduced using many approaches: introducing aesthetically pleasing game graphics, intuitive reward systems or player autonomy, thus enticing students to persist with learning. GBL presents learner instructors an opportunity to provide a more interactive approach to learning which supports both a deductive and inductive learning instruction.

A previous study by Pierfy (1977) investigating the effects of GBL games on students provided very optimistic results. Pierfy (1977) tested student's knowledge retention when playing training games, the results indicated that in eight of 11 cases, knowledge retention was superior in GBL. Additionally, student's motivation was tested when playing training games, and the results conveyed that in seven out of eight cases students showed a greater interest in game activities, in comparison to standard classroom instruction.

Hamari et al. (2016) investigated the impact of flow on user engagement in GBL, and concluded that tasks which required great skill and that correlated with an individual's possessed skill level resulted in deeper concentration and absolute absorption in the task, leaving the individual with no cognitive energy left for distraction. Hamari et al. (2016) also suggests that when the learning curriculum is introduced in GBL, it is more likely to invoke engagement and flow in students.

Killi (2006) stated that by incorporating well structured, cohesive narratives with learning content, the learning material is more easily grasped than when it is presented in a generic decontextualized form. Further research (Cordova & Lepper, 1996)

supported Killi's opinion and explained that material is better learned when presented by environments, characters, and topics that are of high interest value to students.

Player autonomy was determined as another important factor that promotes intrinsic motivation in GBL and as supported by Zuckerman et al. (1978), individuals whom were offered choices and had control over what they were doing conveyed a higher level of enjoyment, and engagement, as well as being more determined in completing activities. As mentioned previously by Malone (1980), motivation can be provoked by introducing knowledge inconsistencies or teasing individuals with intriguing pieces of new knowledge. Motivation can then be maintained by managing an optimal level of informational complexity, not to overload or confuse the individual, but just enough to remain curious. By implementing sensory delight, fantasy, well-aligned goals, unpredictable outcomes, and player autonomy to today's GBL games, it is possible to achieve motivated and engaged students. The use of GBL can be a catalyst for improving one's self-efficacy or self-esteem towards a learning topic presented by the game by making use of the playfulness of the learning environment (Meluso et al., 2012). This self-efficacy can impact the effort put into learning activities, be it game or classroom learning, and having the student take on challenging tasks. As the student has a higher self-efficacy they are also likely to attempt to complete a task even when facing difficult situation.

Scaffolding

As the GBL solution is being developed, an important element to investigate and possibly implement is the use of scaffolding. Scaffolding is a technique where one or multiple external factors support and supplement the game (Barzilai & Blau, 2014). The core feature of scaffolding is its aid in conveying information needed to understand and as such overcome challenges that are presented during the game session. Though the learning through the game might be sufficient, the use of scaffolding can help convert the knowledge into more abstract concepts. The aid is used to connect game and theory, helping the player make connections of his knowledge learned from a specific game environment to other applications, such as merging it with knowledge from school, and understanding the concepts presented. This bridging of knowledge is important as the way it is used in different practices may vary. During game play, the knowledge is used in conjunction with game visuals, mechanics and feedback, which is very different from more formal structures of e.g. classroom learning which makes use of formal language, symbolism and other more abstract representations (Barzilai & Blau, 2014).

Scaffolding can be implemented in a variety of ways. With the game as focus, scaffolding can be inserted as a pre- or post-activity as well as during game play. It is noteworthy that one is not restricted to one scaffold, and as such, implementation in all three phases are possible. Traditionally, scaffolding has been created around teacher-student and peer relations, making use of debriefing and discussion to process the knowledge learned. These methods have gained the company of technical solutions based on software implementations. Such scaffolds could be websites or subsidiary programs to the game. Barzilai and Blau (2014) present two methods to benefit from scaffolding: *structuring* and *problematizing*. To use the scaffold to structure the information, one

must reduce the complexity. This could be done with visual representation and concretizing more abstract definitions into easily understandable terms and forms of usability. Problematizing is the attempt to centre the attention towards critical ideas and connections. This can short-term be difficult for the users to work with, as there is still a degree of complexity in the information, but long-term problematizing should present deeper processing and productive learning experience. Barzilai and Blau further expand on scaffolding by defining flow, presented earlier, and enjoyment ("*a positive reaction to an experience that involves intertwined physiological, affective, and cognitive dimensions"* (Barzilai & Blau, 2014) that may occur from sensory delight, suspense and relief, achievement and self-efficacy) as factors that can be affected by a scaffold. However, their research shows that external scaffolds do not affect the perception of flow and enjoyment. Instead the scaffolds impact users perceived learning from the game, believing what they have learned was not a result of the game itself. This change in perceived learning is greatest when using pre-game scaffolds. The lack of effect on flow and enjoyment may be a result of only using pre- and post-game scaffolds, and it is suggested that scaffolds during the game may have a different impact. Furthermore, the scaffolds were designed to fit the game narrative, which may have led to a perception of the scaffold being part of the game, and as such not affecting flow. The test suggests that the use of scaffolding pre-game has a better effect on learning, as participants had a better post-test assessment. Scaffolding can also be applied to any form of teaching instruction: *inductive*, and *deductive*. Inductive instruction is the method of teaching that presents students with new pieces of knowledge at every learning step, in inductive instruction the student is presented with a number of tasks and activities that infer the new rule being taught. In contrast, deductive instruction is a more efficient method of teaching although there is a lack of new knowledge presented in each step, as the methodology of deductive instruction is to present a rule, and then apply the rules to specific examples and activities that use the rule.

An instructional game model that takes advantage of scaffolding (debriefing) is the Input-Process-Outcome Game Model by Garris et al. (2002). The model illustrates that instructional content is combined with game characteristics to form an instructional game, and that debriefing is an essential process step when learning through instructional games. Garris et al. (2002) describe that users of instructional games experience a cyclical process where they make judgements on their game experience, leading to user actions and motivation, and finally resulting in feedback provided by the instructional game, and based on the received feedback, the user will decide whether to continue with the game experience. Ultimately, the last step of the process is debriefing where the purpose of the step is to bridge the gap between what was learnt in the instructional game, and the individual's learning objectives. This is to ensure that the learner can distinguish between the learning content and the game characteristics, and that the instructional game was beneficial in contributing towards their learning objectives.

Principles of Game Design

Input-Process-Outcome Game Model

Garris et al. (2002) make the obvious statement that instructional games can be created by merging instructional content with game characteristics, but further suggest that game characteristics in instructional games can be categorised by six dimensions: fantasy, rules/goals, sensory stimuli, challenge, mystery, and control.

Fantasy

Incorporating fantasies into instructional games offer students the opportunity to learn real world skills from fictional analogies, and allows learners to fail safely in an environment insulated from real consequences. From previous research Garris et al. (2002) suggest that fantasy can exist in two contexts: *endogenous*; a fantasy that has relevance to the learning content, *exogenous*; a fantasy that is superimposed over the learning content.

Rules/Goals

The rules of a game define the goal structure within the game, and by establishing clear and purposeful goals students are able to self-evaluate their performance against the current game goal, which is crucial in provoking motivated learners whom have a high level of goal commitment. Garris et al. (2002) contemplate on Crookall et al.'s (1987) three rule forms that operate in a game experience: *System rules*; rules that control the operation of the game world, e.g. player movement and game physics. *Procedural rules*; rules based from game events, e.g. when a player reaches level 10, a new level is unlocked. *Imported rules*; rules that players import into the game from the real world; implied rules that govern general behaviour, e.g. you cannot walk through walls.

Sensory Stimuli

As games are set in a fictional world, players temporarily accept the existence of another form of reality. Games use audio, animation, and high fidelity graphics or unique art styles to grab the attention of players.

Challenge

Similar to the concept of *"flow"*, games must possess an optimal level of challenge; goals should be obtainable, and clearly specified. In addition, much like Malone (1980) suggested there must be an optimal level of information ambiguity to ensure outcomes remain uncertain. The combination of embedding learning content in absorbing fantasies can yield purposeful goals.

Mystery

Games that possess the attribute of mystery ensure that individuals remain curious, and thus are motivated to maintain engagement. Garris et al. (2002) refer to previously mentioned theories of diverse and sensory curiosity (completeness, consistency, parsimony) that games must embrace in order to provoke curious learners.

Control

Games must respect player autonomy and allow students to take control of their learning in order to maintain learner motivation. Previous research proved that the freedom of learner control compared with program control yielded higher levels of individual motivation. Garris et al. (2002) further suggest that learner control can be incorporated into a game by allowing players to dictate the outcome of their actions (when possible) or select game strategies. Garris et al. (2002) briefly specified the relationships between objectives of instructional game design, game features that elicit user reaction and feedback, and the achievement of learning objectives in their Input-Process-Outcome Game model (IPOGM) (see figure 9).

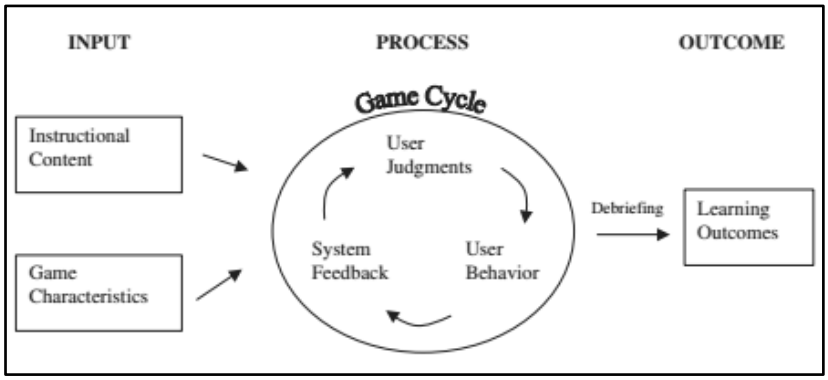


Figure 9 - The Input-Process-Outcome Game Model (Garris et al., 2002).

Input

As the IPOGM conveys, Garris et al. (2002) deconstruct an instructional game into two distinct components: *instructional content*; the learning material, *game characteristics*; the six components mentioned previously. Therefore, the input to the process and game cycle is instructional content that incorporates game characteristics.

Process

The game cycle is initiated with *user judgements*, where individuals generate subjective opinions regarding the game process. Garris et al. (2002) categorize these opinions into four elements of judgement: *interest*; when an individual perceives the game content to be of interest, *enjoyment*; when an individual is perceived to be enjoying and engaging with the game, t*ask involvement*; when an individual becomes absorbed in a game activity, *confidence*; when an individual feels a sense of safety in making mistakes to gain familiarity and build on existing skills. The penultimate step of the game process, after the formation of user judgement and sustained gameplay is the determination of *user behaviour*. At this stage, it is possible to evaluate whether the individual is motivated by their judgements, based on the intensity and quality of the user's actions: becoming more involved with game tasks, pursuing goals, and commitment to task completion. Thus, leading to the concluding step of the cyclical game process; *system feedback*. In this step; as a result of sustained engagement, individual performance is evaluated by feedback from the system. Depending on the consistency of feedback reflecting on the user's performance, the individual may continue to advance back to the first step of the

game process (user judgements), where the individual will generate further opinions regarding their game experience, before engaging yet again.

Outcome

The first output from the instructional game experience is the process of *debriefing*, with the aim of providing a coherent and cohesive educational experience, and bridging the gap between the game cycle and learning outcomes. As instructional games are a form of experiential learning, the purpose of debriefing is to ensure that the fantasy and learning content are not confused, and to ensure there is facilitation and learning support to ensure effective learning. The implementation of debriefing in the IPOGM was influenced by McIntyre and Rubin (1971) processes of *"doing, reflecting, understanding, and applying,"* to support effective learning.

Perhaps the most fundamental component of the IPOGM is *learning outcomes*. To simplify the complex construct of learning, learning outcomes have been divided into three categories, (see figure 10): s*kill based*; learning outcomes that target sensorimotor skills, e.g. training aviation skills using a flight simulator. *cognitive*; learning outcomes that address knowledge structures, which is further deconstructed into three sub-components: *declarative knowledge*; to replicate or remember an item of information, *procedural knowledge*; applying knowledge or skills to a specific problem, *strategic knowledge*; applying learnt principles and theories in different contexts to relevant situations. a*ffect*; learning outcomes that specify the attitude of the individual, e.g. to promote feelings of confidence and self-efficacy.

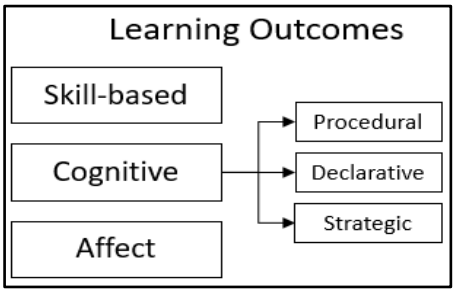


Figure 10 - Conceptualised learning outcomes in the IPOGM.

The IPOGM emphasises on learning via active engagement within the game's environment, and by incorporating this methodology with the process of debriefing, instructional games can provide a more effective learning environment.

Heuristics for Instructional Game Design

Malone (1980) suggests a set of heuristic instructional game guidelines which can be used when designing and developing educational games that incorporate GBL. Malone (1980) defines three principles to consider when designing intrinsically motivating environments in games: *challenge*, *fantasy* and *curiosity*.

Challenge

An instructional game has the quality of being challenging when appropriate goals are introduced and outcomes are unpredictable.

Goals

- Goals must be distinct.
    - Although goal strategy should remain unpredictable, yet coherent.
- Goals must be attractive
    - When possible, the game can make goals more desirable by providing compelling visuals or incorporating them into an enticing narrative context. For example, calculating how much fuel a rocket needs to make it possible to fly to the moon, from earth.
- Goal progress
    - The game should be able to notify and update the learner on their performance and progress they are making towards the game goal.
- Progressive linear difficulty
    - Progressive difficulties make it easier for the learner to achieve goals. Failure to respect this may result in learners facing discouragingly difficult tasks, quickly resulting in a lack of interest to continue with the learning experience.

Unpredictable Outcomes

- Varying levels of difficulty
    - Game difficulty is generated based on the performance of the player.
    - Game difficulty is chosen by the player (easy, medium, and hard).
- Multiple goals
    - Score-keeping: the score can reflect the number of attempts, items collected and difficulty of success.
    - Speeded responses: to do something as fast as possible.
- Hidden information
    - Hiding information provokes curiosity in players and applies an aspect of challenge to the game.
- Pure randomness
    - An approach to introduce definite uncertainty is randomness; unpredictable by human or machine, forcing game outcomes to be completely randomised every time.

Fantasy

Encapsulating learning content in a compelling fantasy makes the game experience more engaging, and by introducing fantasies into learning contexts, it may support learners to become more absorbed in the learning experience. However, an obvious issue with fantasy is that some people prefer different fantasies over others, and thus not everyone will appreciate or be motivated by the same genre of fantasy. Malone (1980)

then suggests a notion similar to that of Garris et al.'s (2002) exogenous and endogenous fantasy contexts, and exemplifies that fantasies can exist in intrinsic or extrinsic form, and claims that intrinsic fantasies are better in provoking intrinsic motivation. As figure 11 illustrates, extrinsic fantasies depend on the use of the skill that the game is teaching, and whether it is being used correctly (answering questions correctly or incorrectly). Although, what is common amongst all extrinsic fantasies is that the exercising of the skill does not depend on the fantasy in anyway (influencing the fantasy world). In contrast, skills in intrinsic fantasies do depend on the fantasy as well as the fantasy depending on the skill. This means that typically the problems which test the skill are conveyed and made relevant to the fantasy world, and the performance of the player has an impact on the state of the fantasy world.

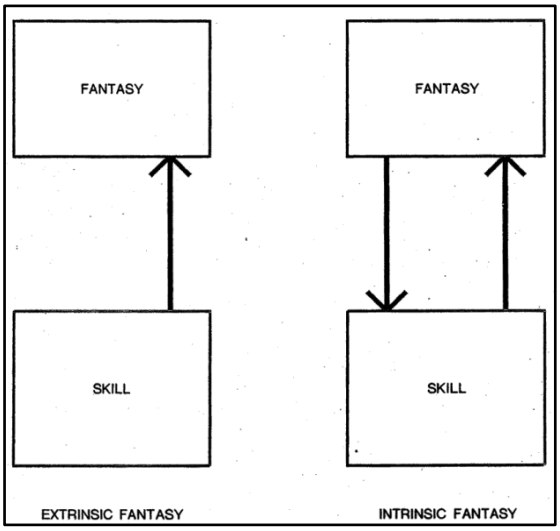


Figure 11 - Comparing skill and fantasy dependencies in intrinsic and extrinsic fantasies (Malone, 1980).

Intrinsic

Intrinsic fantasies arouse a situation where the skill being learnt may actually be used, although the situation does not have to be a pragmatic example. For example, in a fantasy world the student may be put in the role of a carpenter who manufactures rectangular tables for his customers. Each of the carpenter's customers supply him with a piece of a rectangular wooden material that they wish for him to carve a table from, thus from this fantasy situation the player can measure the width and length of the piece of wood, and from the known dimensions they are able to calculate the area of the rectangle. By using the skill of calculating the area of a rectangle correctly the fantasy depicts a happy customer, whilst on the other hand using the skill incorrectly results in unsatisfied customers.

Extrinsic

Extrinsic fantasies are fantasies that are overlaid on top of the learning content, where the skill of the student has an impact on the state of the fantasy, although the skills being used in the fantasy are not necessarily being learnt through the fantasy itself. For example, students may attempt to correctly answer arithmetic related questions to save a princess from an evil king. Depending on the goal of the narrative, the goal of the player in relation to the fantasy may be either to: reach a *fantasy goal* or avoid a *fantasy*

catastrophe (see figure 12). However, the disadvantage of implementing a fantasy catastrophe is that players may be tempted to answer incorrectly just to be able to witness the catastrophe.

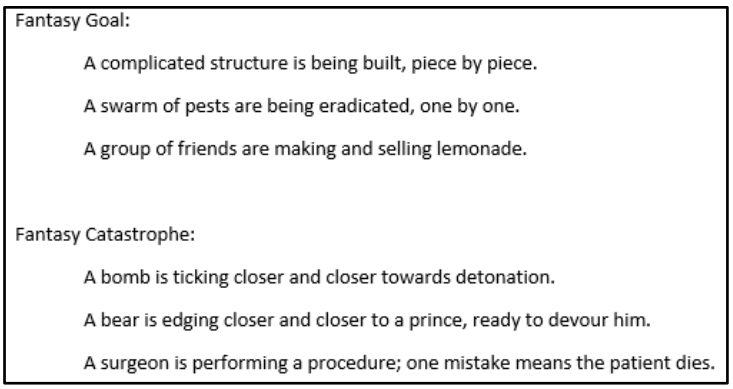


Figure 12 - Comparing fantasy goal with fantasy catastrophe.

Curiosity

Malone (1980) describes *surprising feedback* and *constructive feedback* as *informative feedback*, and furthermore indicates that informative feedback is a means to make environments increasingly intriguing and complex:

Surprising Feedback

To sustain curiosity, feedback should be unexpected and random, although the information should be kept consistent in order to maintain coherence of the game's fantasy.

Constructive Feedback

To remain educational, feedback should not just be a simple statement of correctness or incorrectness. Instead, the game should explain to the player why they were incorrect so it is possible for the player to synthesise a coherent understanding and reasoning for the correct answer, and thus allowing them to update their knowledge structures with logical information via reflective learning. Moreover, Malone (1980) illustrates that curiosity can be provoked in virtual environments via the use of compelling audio and graphics assets:

As decoration

Audio-visual assets that are naturally a part of the game scene; regardless of player interactivity. These assets are crucial for establishing a positive first impression, and are important for establishing an initial interest in the game.

To enhance fantasy

Audio-visual assets that contribute to, and enhance the fantasy that exists within the game. Special effects generate a more captivating experience, and by merging such effects with the game's fantasy, the player's perception of the fantasy can be intensified, i.e. implementing particle systems to augment spell-like visuals.

As reward

Audio-visual representations implemented to honour good performance which can escalate the importance of the goal. Although, Malone (1980) expresses that the activity of pursuing a goal as a purpose to retrieve a reward can undermine people's curiosity.

As a representation system

To represent game information more intuitively; using informative graphical representations rather than textual representations. For example, in GTA V a player that is swimming underwater can determine whether they are drowning via audio-visual feedback: *audio*; obvious groans of struggling for air, and *visual*; the screen gradually becomes darker and darker indicating their life flashing before their eyes. Therefore, by implementing Malone's (1980) insights and whilst respecting his philosophy of informational complexity, fantasy, challenge and curiosity, it is possible to design an intrinsically motivated virtual environment using a heuristic design approach.

Designing for Children

As this report investigates cooperative and GBL elements on children, it is important to know how an ideal solution for a children's game can be designed, so that the game itself does not take away from the experience by introducing technical issues and problems. This design for children further helps enhancing the game experience, giving a more realistic picture of motivation and technical challenges for other eventual productions for education and learning games.

Chiasson and Gutwin (2005) have attempted to create a catalogue of design principles that reflect upon on child development. Here they stress the need for designing specifically for children as their development process has not yet reached the state of adults, and as such a solution for adults cannot be translated directly to a children's version without issues. They further instruct that even amongst children there can be a variation in development, and the design should be focused on smaller segments of age groups. The catalogue presents three types of children's development that must be addressed in the design process:

- Cognitive development
- Physical development
- Social development

For the children's cognitive development, the designers must find solutions that makes use of visuals and audio rather than text, as children may not be able to read or have the same understanding as adults e.g. in relation to metaphors or higher level wording. The use of simple visual metaphors can however be used as a substitute for text, having the visual being easily relatable to a concept. This type of visual metaphor could be the image of an animal instead of its written name. As children are not necessarily used to reading on a screen, the designer could opt for auditory aid; reading texts out loud or using audio only. The children's mental development regards the ability to understand abstract concepts. Children may not be able to understand abstract information and interfaces; thus, the design should reflect this by being simple. During use of the

solution, children will learn through trial and error, mapping cause and effect of their actions. The higher the degree of abstractness, the harder this mapping can be. Another important factor for children to understand and work with a solution successfully is feedback. Children expect immediate results to their actions, and may retry actions if feedback was not presented. As mentioned with the children's ability to read, instructions must be intuitive or the solution must guide the children during the execution of tasks. As presented earlier in relation to GBL the use of scaffolding could be a way to implement these guiding features. Complexity of tasks should be simple, and can then increase in difficulty as the children learn to work with the solution. Feedback can in this case be quite heavy in the beginning, and then be gradually removed as the child improves. This encourages the child to take more cognitive responsibility. The use of feedback should further be designed so that children understand why the feedback was given, giving a clear relation between action, result and feedback. The last factor Chiasson and Gutwin (2005) present as part of cognitive development is the child's imagination. Children may understand a situation present in the solution as one that reflects real life. If a concept is learned from real life, they expect the concept to work the same way in the solution.

Regarding the children's physical development, design must take into consideration the motor skills of the children. Adults are better at fine motoric actions, whereas children have more crude motions. Working with computers, the user can make use of keyboard and mouse as traditional physical interfaces. For children, it can be hard to use the keyboard, as they search for the needed button in a time-consuming hunt-and-peck strategy (Chiasson & Gutwin, 2005). Using the mouse can also be an issue, as clicking and holding a button while moving the mouse can be challenging. Drag-and-drop type circumstances can be troublesome to complete efficiently. To counter this, one can attach the clicked element to the mouse as to remove the need of holding it down. Children can also have issues with clicking smaller objects on the screen. As their motions are crude, smaller adjustments of mouse position may not be fine enough, and then the child cannot click its intended target.

Chiasson and Gutwin (2005) separate the social development into three subsections: motivation and engagement, social interaction and collaboration. They suggest that motivation and engagement can be implemented by creating the feeling of empowerment in the children. They can be given control of the environment and decide the speed at which they should operate. Tying into the cognitive development, this power teaches the children about consequences to their actions. On-screen characters can also be a motivating and engaging factor. These characters can be used and instructions, giving explanations, while the children find them interesting. Lastly a motivating factor can be extrinsic rewards as presented earlier. Chiasson and Gutwin (2005) suggest to use this extrinsic motivation as an addition to intrinsically motivating designed rewards. These extrinsic rewards can be anything such as traditional scoring systems, messages and bonus activities. The availability of social interaction is expected by children if the solution includes online features because other solutions make use of social interaction when online features exist. However, children can still have a hard time opening up and participate socially (they might be anything from shy

to unable to interact by traditional means). To help with this, a design that introduces a layer of anonymity can to a degree help breaking down the barrier. Finally, collaboration is presented as a cooperative feature where children group up and learn together. This could be by sharing a computer, working together in completing the challenge on the screen. This however introduces the issue of loss of interest. This will happen to the passive participants (the ones not controlling the mouse and sitting in front of the screen) first. A solution to this could be online cooperation allowing every individual to have a screen and control.

Narratives in Game Based Learning

As Garris et al. (2002) suggest, implementing a provocative narrative that students can relate to in the learning content can be a useful motive to initiate and sustain learning. There are several factors to consider when aspiring to incorporate a cohesive narrative in GBL; the need for interactivity in DGBL introduces another dimension of complexity to the activity of designing and implementing a narrative, as player interaction introduces the problem of contradicting the structured narrative with unpredictable interactivity. As the system has two dimensions in narrative and interaction, one must define a goal of how the two dimensions connect. Bruni & Baceviciute (2013) presents three types of goal structures. First is a combination of the narrative and system goals, secondly the narrative goals is subsidiary to the goals of the system and third the narrative goals are nonessential to achieve system goals.

Narrative Structure

The structure of a narrative depends on the goal of the system, and requires a careful balance between narrative coherency and interactivity. Therefore, the objective of the narrative defines the structure of the narrative: High narrative coherence & low interactivity; to facilitate a rich, coherent narrative, Low narrative coherence & high interactivity; to facilitate a robust narrative that maintains coherency when introducing autonomous player interactivity.

As mentioned previously, engaging narratives account for a high degree of agency and interactivity in order to ensure the user is able to direct the discourse of the narrative in multiple directions. To accommodate for multiple narrative outcomes, the narrative structure requires many more nodes, pathways and closures, which inevitably leads to many possible outcomes, resulting in *combinatorial explosion*.

Ryan (2001) explains that there are three distinct narrative themes in which to represent a temporal sequence of story events: *Sequential narrative;* a representation of events in chronological sequence, *Causal narrative*; a development from equilibrium to crisis and back to a new form of equilibrium, *Dramatic narrative*; a rise and decline in tension and narrative suspense. Furthermore, Ryan (2001) illustrated nine distinct narrative structures with varying levels of user agency, intelligibility, narrative flexibility, and vulnerability to combinatorial explosion. However, only three structures are identified as relevant to our target narrative motif, and thus are explicitly evaluated.

The Tree

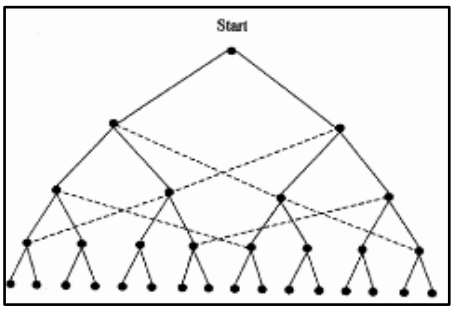


Figure 13 - The Tree structure.

The tree structure (see figure 13) does not support a circuit structure, making it impossible to experience all narrative outcomes in one play through. Therefore, users must play through the narrative multiple times in order to gather all narrative information. The tree structure supports a high level of narrative agency and because of this the dimensional structure is large, and can quickly result in combinatorial explosion.

The Flowchart

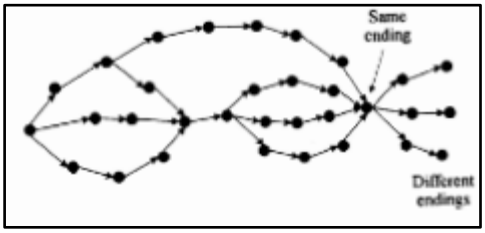


Figure 14 - The Flowchart structure.

The flowchart structure (see figure 14) proceeds through the narrative in a chronological manner and much like the tree structure, eliminates the issue of the user running around in circles. Consequently, the structure does not allow the user to backtrack to previous nodes to verify narrative information. The flowchart is an adaptation of the tree structure which offers a lower level of narrative agency but advantageously is resistant to combinatorial explosion. The structure also represents a very repetitive penultimate conclusion but advantageously offers more than one option for narrative closure.

The Vector with Side Branches

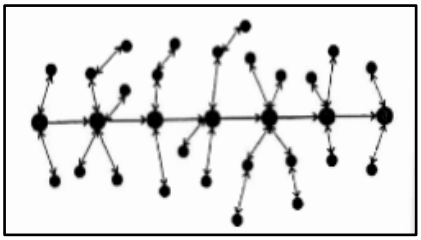


Figure 15 - The Vector with Side Branches structure.

The vector with side branches (see figure 15) advances through the narrative in a relatively sequential manner, whilst also accommodating for some level of narrative agency. The structure allows the user to take small detours away from the main narrative discourse, to indulge in other experiences; whilst still offering the user to return to the main narrative at any given time. Should the user choose to explore a branch in the vector, it may result in an increased level of narrative intelligibility, although not affecting the state of the story. Due to the simple architecture of the structure it is immune to combinatorial explosion.

Narrative Interactivity

The activity of incorporating narrative interactivity with a narrative introduces the common issue of attempting to negate the *narrative paradox*. The narrative paradox is introduced when player interactivity contradicts the nature of the narrative; normally occurring when some overly curious individuals experiment with what impact their bizarre interactions have on the overall cohesiveness and coherence of the narrative. Therefore, accommodating for a higher degree of interactivity autonomy normally introduces a greater risk of inducing the narrative paradox.

Ryan (2001) formed a framework for narrative interactivity based on user intentions and autonomy (M.-L. Ryan, 2001):

- **Reactive interaction** - No deliberate action on the part of the user.
- **Random selection** - The user takes action, but cannot foresee the consequences.
- **Purposeful selective interactivity** - The user can choose between two different story paths.
- **Productive interactivity** - The user is incorporated into the narrative world, and actions taken, influence the narrative.

In later work, Ryan (2006) nominates four forms of narrative interactivity based on two dichotomous values: mode; *internal*/*external*, and impact; *exploratory*/*ontological*. Ryan describes an internal mode as when the user can identify themselves as a character in the virtual environment. In contrast, she describes an external mode as when the user is assigned to an observer role, where they perceive them self as a god-like figure in the virtual environment. An exploratory impact is represented when the user has no impact on the destiny of the virtual environment, whereas an ontological impact defines that the user has a transformative effect on the game world. Therefore, a taxonomy of these two variables results in four forms of interactivity: *external-exploratory*; e.g. reading a book, *internal-exploratory*; e.g. environmental storytelling, *external-ontological*; e.g. simulation games, i*nternal-ontological*; e.g. adventure games.

Narrative Intelligibility and Closure

To ensure students are motivated to attain their target learning objectives, narratives can be incorporated into GBL to provide more engaging learning contexts. Although, it is paramount to ensure that the full attention of students is not diverted towards the understanding of the narrative, but instead focused on completing the challenges and

tasks supported by the narrative, thus facilitating learning and the accomplishment of learning objectives.

Bruni and Baceviciute (2013) express narrative understanding as the concept of the *author-audience distance (AAD), narrative intelligibility* and *narrative closure*. AAD refers to the interpretation gap between the audience's understanding and the author's meaning and intent in the narrative (see figure 16) (Bruni & Baceviciute, 2013). When presented with an abstract narrative this interpretation gap will be bigger than when presented with a didascalic narrative containing clear relations and a story that is perceived as it represented. When introducing interaction to the narrative, there is a possibility of increasing the interpretation gap, as the narrative allow for a user's intent and as such forgo the author's intent. The author's intent is however not necessarily disregarded as these can be embedded in the rules and boundaries of the system.

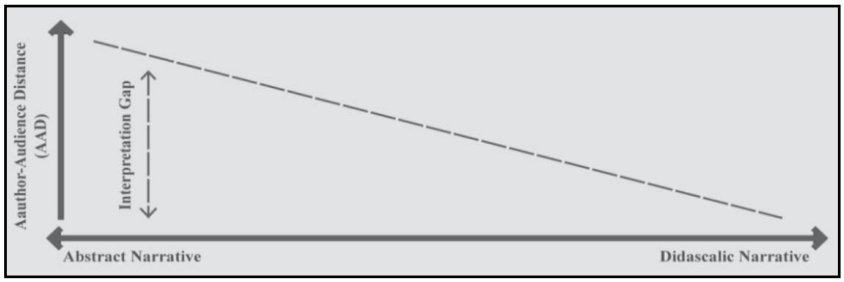


Figure 16 - The author-audience distance model (Bruni & Baceviciute, 2013).

Narrative intelligibility refers to the user's ability to generate an interpretation of the narrative that is close to the author's intentions and expectations. The AAD model can be interpreted as a function of narrative intelligibility. Narrative closure also refers to the ability to generate a meaning from narrative events. In contrast to narrative intelligibility, narrative closure does not require a relation to the author's intent. Therefore, it is solely dependent on the user's capacity to generate a satisfactory narrative closure.

Application - Plain Probability

Design

Conceptualizing the world

To create a fitting game for the test procedure of cooperation in a DGBL environment focusing on mathematics in probability, we had to design a game fitting for the test demographic taking into consideration the theories presented in related work. The first steps taken were to decide on the genre and style of the game (Plain Probability) to ensure a uniform agreement of the design and development process, keeping consistency. As presented by Malone (1980) and Garris et al. (2002) the use of a fantasy, or fictional world can be an effective means to offer a safe learning environment. We worked on designing a fantasy setting for the game to unfold in, where the use of mathematics would become an integral part of the fantasy. To create this fantasy setting

we decided to use the fantasy genre designing with the idea of possibly adding exotic elements such as wizardry and other traditional elements of fantasy which could be interesting for the children.

A decision was made to have the game's mood (see figure 17) and style light, with nice clear colours and lower polygon count objects in the environment. However, the design of the instructional game was more heavily influenced towards the accomplishment of the experimental goals rather than an impressive aesthetic appeal. To further incite motivation and attention towards the game, we chose to incorporate character animation to support sensory stimuli, which as presented by Garris et al. (2002) could persuade players to accept the game world as a form of reality. It was determined that either a 3D (with fixed camera position) or isometric graphical representation would complement the representation of the game environment and game mechanics.

Figure 17 - Mood board for Plain Probability.

To find a matching challenge level to the test demographic, we conferred with a set of mathematics learning books for third graders: *Format 3 Elevbog* and *KonteXt 3 Elevbog* (Madsen et al., 2008; Jensen et al., 2006). Besides showing the level of challenge, these books showed the presentation and format of mathematics questions at school, which was a great help to design questions with strong enough visuals in more concrete situations.

Narrative

A fantastical narrative was implemented into Plain Probability as one of few motives to drive motivated learners, and attempt to make the activity of learning mathematical probability more stimulating and intrinsically motivating. A Vector with Side Branches narrative structure was initially considered, to allow players some degree of narrative agency and have some influence on the narrative progress; another reason for the consideration was to allow players to indulge in further narrative experiences, possibly resulting in an increased level of narrative intelligibility. Although, a fundamental

reason for implementing a sequential narrative structure was to give students a motive to keep on progressing linearly through the narrative and to encourage them to keep on learning about probability. The narrative remained simplistic and didascalic in nature leaving less room for inaccurate interpretations of the narrative, and thus increasing the likelihood of the player maintaining a high degree of narrative intelligibility and closure, which was the goal of the author (us). Another reason for implementing a simple narrative was to ensure that the student's concentration was not diverted away from the learning content, as the goal of the narrative was classified as being subsidiary to the other existing goals of the system.

After specifying the narrative structure the mode of narrative interactivity was determined, based on Ryan's (2001) initial taxonomy of narrative interactivity, the mode of interactivity which exists in Plain Probability can be defined as the type *Productive Interactivity*, as the player is introduced into a narrative world where their actions have an influence on the narrative. Moreover, based on later work by Ryan (2006), the form of interactivity that exists in Plain Probability can be described as *Internal-Ontological*; as the player can identify themselves as a character that has a causal effect on the narrative, and that the narrative itself has a transformative effect on the game world, e.g. building a residential house, helping the farmer grow his first set of crops in his allotment.

The narrative plot in short; a town mayor of a small town located in Emerald Valley requires the help of a mathematical genius to guide him in making appropriate decisions that positively impacts the town's best interests of growth and sustainability. The player is firstly introduced as a potential candidate for the position, whom after proving themselves is appointed the role of a mathematical genius. However, the town mayor has an envious assistant whom at first strongly dislikes the player, due to the fast-progressing, close relationship being established between the town mayor and the player. Therefore, the player must passively deal with this complication whilst also doing his job, assisting the town mayor as well as other citizens with complex mathematical errands of probability. The narrative encapsulates the learning content in a fantasy setting, where the fantasy is very much relevant to the skill being learnt (mathematical probability). The rationale for incorporating the narrative and learning content in this manner was to respect Cordova & Lepper's (1996) statement of *"instructional content that is embedded in fantasy contexts leads to greater student interest and increased learning"*, and thus increasing the likelihood of providing a more motivated learning environment. The narrative gives a reason for probability to exist in the fantasy, and presents the player with several situations where the need to understand probability is vital, e.g. by accident, the farmer has mixed all his seeds into one bag, he knows how many of what seed exists in the bag but he cannot tell the visual difference between each seed. Therefore, the farmer needs guidance from the player to help him understand what the chance is of him pulling out a specific type of seed to sow. An example of the question that exists in the game "What is the chance of the farmer pulling out a carrot seed, if he knows he has 25 carrot seeds and there are 50 seeds in the bag." By incorporating the narrative in this manner, we respect Malone's (1980) insight of intrinsic fantasies being more likely to provoke intrinsic motivation, as the fate of the fantasy depends on the

player's skill (answering a question advances the narrative), and the use of the player's skill is relevant to the fantasy world (the player's use of skill can be applied to situations in the fantasy).

According to Garris et al.'s (2002) taxonomy of fantasies with his statement of *"endogenous fantasies are more effective motivational tools,"* and the subsequent description of Plain Probability's fantasy, it is safe to assume that the narrative can be classified as supporting an endogenous fantasy, as the fantasy is directly related to the learning content. Furthermore, Plain Probability's narrative implements constant positive feedback as motivation and reinforcement that the player is performing well based on their interactions; even if the player answers questions incorrectly the discourse of the narrative was not affected, and positive feedback is shared regardless. Therefore, Plain Probability's narrative design ideology aligns with multiple beliefs and theories: Deci and Ryan's (2000) belief of when trying to make an activity intrinsically motivating, feedback should be presented positively, as negative feedback may inhibit intrinsic motivation for the current activity. Garris et al. (2002) suggest that feedback is an important component of the cyclical game process, as a positive interpretation of feedback is more likely to persuade the player in making further user judgement, resulting in a repeat of the process cycle, and Fisher (1978) states that by granting player autonomy, individuals are able to be responsible for their own performance as a method to provoke intrinsic motivation; this was achieved in Plain Probability by allowing players to select their own answer resulting in success or consequence of their own actions.

Characters

To help create the atmosphere of a living world and a fantasy that could be a reality, we designed the game to make use of NPCs, having the players interact with them to obtain and finish goals in the game, having the player included in the narrative through the NPC interactions. The NPCs would work as a tool to induce immersion, as Ermi and Mäyrä (2005) present can be done with imaginative immersion. Using NPCs allowed for giving instructions in-game, letting the player focus his attention towards the game. This instruction presentation was chosen to possibly induce engagement with the world and ensuring a sense of flow. Furthermore, NPCs can be used as an interesting feature, motivating through the curiosity of what the character might do the next time an interaction happens. This could in an extended design e.g. the NPC telling a joke, or performing a unique animation, etc.

Level Design

As the game promotes exploration and learning, we designed the world to make use of a centre point; a quest-hub. Here player would initiate their exploration activities. To encapsulate the world, as it is exploratory but not infinitely big, we set up a natural looking border. This border was placed far away enough from the game action as to not reduce the feeling of an open world, yet close enough to ensure the world was not too big for the player to get lost in. The world was designed to feature diverse environments such as a town, forest and stone quarry; replicating the perception of a realistic fantasy world. This environment required a coherent pathing structure needed to ensure that

players knew their location and could communicate that information in a multiplayer experience (see figure 18).

Figure 18 - Map of game level.

The quest-hub was designed to support internal-ontological narrative interaction (Ryan, 2006), by introducing a location where the player could build upon and upgrade on existing objects in the game world. The NPCs were placed here and by the outskirts, making the player know where to find them and handing in missions. To promote exploration, the items to be found were placed on the outskirts and further out, having the player move away from the hub, seeing the rest of the environment. One item was placed in town as a means of in-game introduction to item UI. To fit the narrative and fantasy, the items were placed in environments congruent to their material and purpose.

Gameplay

Movement and Interaction

The peripherals used for Plain Probability were considered to be exclusively mouse/keyboard, complementing either a click to move control scheme or one that utilised W, A, S, D for movement and mouse click for interaction. However, the idea to use mouse and keyboard was discarded swiftly, and the option was to proceed with a point to click control scheme, due to its advantageously simplistic nature. By implementing a point to click control scheme, children may then become more quickly accustomed to the game controls, giving them the opportunity to enjoy the GBL experience and focus on the learning content, rather than wasting time trying to become familiar with a more complex control scheme. The point-to-click control scheme incorporated universal interactivity actions such as: click to move, click to select, and click to interact. To support children's unrefined motor skills and in some cases lack of patience, all interactable objects within the game world were equipped with a large interactable surface area to ensure that aggressively-impatient or inaccurate clicking of objects did not result in a frustrating experience when trying to interact with smaller objects.

When a player interacted with an NPC, either some or all following game events occurred: updating of quest log, enabling outline highlight of interactable objects, updating of quest icons in game space, narrative text displayed in the lower region of

the screen. In a similar situation, when a player interacted with an interactable object the following game events occurred: updating of quest log, updating of quest icons in game space, disabling the interactable game object, updating the inventory with the game object's sprite.

The camera was set to follow the player from a fixed position. This was done to ensure there would be no issues with having to realign camera angles to see what one is doing and where one is going. Removing the possibility of these issues should allow the player to enjoy the game, without disruptions reducing his immersion. The decision to fix the position of the camera was to keep the player closer to the quest-hub and nearby areas, as the outer areas of the environment was not populated.

Multiplayer

Two main factors were discussed to ensure a good multiplayer experience: communication and shared game-play. Regarding communication, we had the option of facilitating voice communication or text messages. The choice was made to make use of voice communication as this feature allows for quick and easy communication removing the need for a message interface, simplifying controls for the children who, as presented by Chiasson and Gutwin (2005), can have issues with using the keyboard if needing to be quick and efficient, as they are not as fully developed as an adult.

Regarding shared game-play, the discussion was on how many of one player's actions that should influence the others. To promote cooperation, the game state was shared between players: the active mission, inventory, and answers to questions. The alternative of not sharing game-play could lead to a multiplayer game session where the two players are in the same world, but not interacting with each other at all. There was a risk of sharing game-play that one of the players might begin to free-ride or social loafing as the other completed the game's goals, however sharing and having them cooperate can work as a sort of scaffolding where they peer-to-peer help each other getting better and understanding the challenges presented. As a visual aid for when the players were to answer mathematics questions, we designed the possible answers to show an outline, making it easier for a player to see the action of the other.

Reward System

A reward system that supports intrinsic and extrinsic goals is a fundamental component of an instructional game. As mentioned in Garris et al. (2002) IPOGM a positive and motivated user behaviour is a result of the game process reciprocating positive feedback to the player, normally in the form of a goal status update or reward.

Tsai-Sun and Wang (2011), describe game rewards as existing in eight forms: *score*, numbers are used to represent a player's performance, e.g. high scores and ELO rating. *progression*; experience point systems and levelling up, *item acquisition*; acquiring an item as a result of luck, skill or hard work, r*esources*; the amassing of a large number of items, resulting in the feeling of progress and achievement, *achievements*; after fulfilling a game condition an achievement is displayed, *feedback messages*; an immediate "reward", *plot animation and progression*; as a result of completing a narrative milestone, the destiny of the fantasy world is impacted, *unlocking mechanism*; unlocking new

items, locked items can evoke curiosity in individuals. Similarly, Schoenau-Fog (2011) refers to three reward forms: *achievements*; accessing new game areas, unlocking new items and abilities, acquiring new items, *progression;* player progression, experience point systems, unlocking new abilities, mastering a game, *completion*; accomplishing achievements, completing objectives, survival, achieving game completion. Tsai-Sun and Wang (2011) then further develops the idea of how players utilise game rewards to achieve four unique situations on two axes (see figure 19): *social*; ranging from self to others, *effort;* ranging from casual to progress.

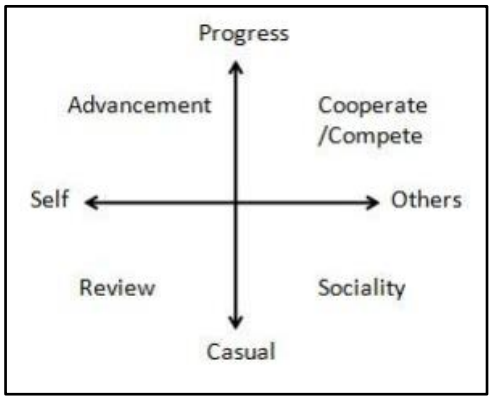


Figure 19 - How players utilise game rewards (Tsai-Sun & Wang, 2011).

- **Advancement** - Using rewards to make progress, growing in increased skill and power. Makes the game fun if the player feels they are improving.
- **Review** - Checking achievement collections, items worn by avatar, watching game cut scenes; making the player feel accomplished.
- **Sociality** - Sharing information about rewards, achievements, powerful weapons. The player wants to establish a social status.
- **Cooperate/Compete** - Sharing resources and hoarding powerful items to maintain advantages over other players/enemies

The design of Plain Probability's reward systems was influenced by considering the different reward forms presented by Schoenau-Fog (2011) and Tsai-Sun and Wang (2011), in addition to Tsai-Sun and Wang interpretation of how players utilise different reward forms. From figure 19 it can be determined that *advancement* is perhaps the most prevalent effect of a reward, and in a cooperative setting *cooperate/compete* is an important reward effect for participants who are motivated by team activities of formulating strategies to overcome team challenges. Lastly, *sociality* may also be an important reward affect for individuals whom are more socially inclined, and are motivated if they are aware that their performances will be recognised and have a social impact.

An initial idea for Plain Probability was to introduce a secondary currency system where after completing a quest successfully, the player would be rewarded with a gold coin which could be used to buy "cheats," where a cheat (50/50) could be used to help the player with future math questions they may face, thus presenting the player with a novel advantage. Another idea was to implement badges as rewards to distinguish between

good performances and exceptional performances, so players whom executed a perfect performance when helping the farmer would be awarded a golden maize badge, and for an individual whom was successful in helping the farmer but did not execute the task perfectly only achieved a maize badge. However, in the final design revision of Plain Probability, players were rewarded with gold stars when correctly answering a probability question, introducing a competitive and cooperative aspect to rewards, as incorrect answers resulted in a consequence of lost score, also supporting Deci and Ryan's (2000) statement of immediate feedback supporting self-determined learners.

Players were primarily motivated by Schoenau-Fog's (2011) "experiencing" and Tsai-Sun and Wang's (2011) "plot and animation progression," as player interactivity and performance had an impact on the progression and the fate of the virtual environment. An example of this; after helping the town mayor establish which material was most optimal when building the town's first residential house, the player then gathered the materials, and finally constructed the house, resulting in the achievement of a major narrative milestone. Another example; when the player helped, the farmer remembers the exact contents of his seed pouch, the farmer could sow the appropriate seeds in his allotment. After finishing the game and completing the narrative, the player was presented with a final score screen where the player could decipher how well they performed based on the number of gold stars and mega stars obtained for each quest line (see appendix A2).

User Interface

The UI is an important element of a DGBL experience and is occasionally undervalued when designing for instructional games. Clever use of UI elements can contribute to both, a more engaging narrative and game experience; made possible by considering more modernised UI design paradigms of: *diegetic, spatial* and *meta* UI elements, and displaying distinct game goals in unison with the player's current progress towards them, respectively. As mentioned previously, UI elements can be divided into two categorical representations of interfacing: 3D space representations and non-3D space representations, which can be further defined as four unique modes of interfacing (Andrews, 2010) (see figure 20): *diegetic*; a UI component that exists in 3D game space and very much supports the game's fantasy, non-*diegetic*; a UI component that does not exist in 3D game space, instead it is rendered on a 2D canvas that overlays the game and does not necessarily contribute to the game's fantasy, as it has the sole purpose of being informational, s*patial;* a UI component that exists in 3D game space, but does not support the game's fiction due to its informational purpose. *meta*; a UI component that does not exist in 3D game space, it is rendered in 2D space that overlays the game, however it does contribute to the game's fantasy.

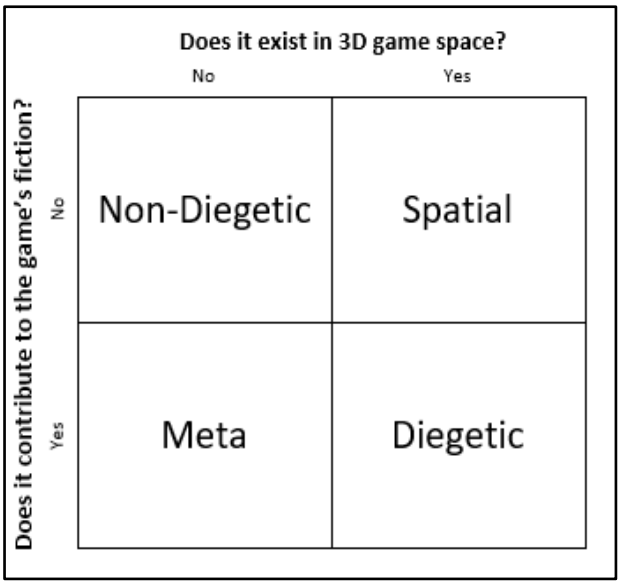


Figure 20 - Four graphical representations of UI components, split up into four quadrants based on their form of existence and contribution towards the game's fantasy.

The conceptualisation and design process of the UI and UI components were influenced by this design methodology and the target population whom Plain Probability was intended for (8 - 10 year old students).

In the multiplayer game build, player 1 and player 2 were depicted using two UI representations: *diegetically*; the character appearance of player 1 and 2 were different by varying the material colour of specific materials applied to each model, and *spatially*; via a colour coded ring that existed around the player's avatar, player 1 was represented as blue and player 2 was represented as red. Whereas generically, quest icons were represented spatially as they had no relevance to the fantasy, but were only displayed to inform the player of the state of the current quest; quest icons (in addition to other game elements) were animated in order to abide by Rieber's (1991) statement of animated graphics enhancing the *"motivational appeal of instructional activities,"* thus providing a more motivating virtual environment. Game objects within the scene utilised an orange outline when it was relevant for the player to interact with object (helping the town mayor find the blueprints), making it easier for players to visually understand the importance of the specific object at a given time.

Information that was important to the player was displayed in a non-diegetic manner: Quest Log; the game's current quest and quest objective, number of Stars; number of correctly answered questions, quest dialogue; the narrative that provides positive feedback and meaning for embarking on the current quest, to ensure that such crucial information was made clear and less likely to be misinterpreted. Furthermore, a display time function for the narrative text was implemented based on how many characters existed in the narrative dialogue string, thus allowing children whom had a harder time reading the opportunity to grasp the narrative as well. The probability questions that were asked by NPCs were answered on a blackboard that existed in the game world, and was referenced regularly in the narrative. Therefore, as the questions were conveyed in 2D space on the blackboard it could be argued that these questions were meta-

representations of UI components, as the UI component contributed to the fiction of the game world. Moreover, when a user selected an answer, it was highlighted to provide obvious visual feedback. In a multiplayer setting, selecting an answer would select the answer for both players, thus forcing cooperation and agreement between players, and on pressing the submit button the answer would be saved for both players collectively.

Implementation

During implementation, we were presented with several challenges. Firstly, we knew that a multiplayer feature must exist in order to facilitate cooperation, thus we utilised Photon Unity Network's (PUN) client server network architecture to establish rooms and lobbies. To accommodate for the need for multiplayer we created a network manager to handle the server, while synchronizing player data using serialization. Player actions (moving and interacting) were achieved by using pointer click event triggers on the *NavMesh* environment and interactable items. All event triggers had to be customised and set up in real time as all players were not included in the scene hierarchy for reference. Furthermore, we implemented an easily extensible quest structure utlising abstract classes and inheritance.

Quest states and game states were handled by conditions and reactions. This idea originated from the Unity "Adventure Game Tutorial" (Unity, 2016). Conditions were *ScriptableObject* types containing a Boolean value to verify fulfilment of the condition, as well as a description string and an integer value. *AllConditions* is a script holding a list of conditions as well as the method to check for condition fulfilment (see appendix B1). Reactions are objects inheriting from the *Reaction* abstract class inheriting from a *ScriptableObject*. Concrete types of reactions, such as *ChangeQuestStateReaction* and *ShowUIElementReaction* utilised the *React()* and *ImmediateReaction()* method to execute all encoded actions when activated. To hold these conditions and reactions, a *ConditionCollection* (see appendix B2) and *ReactionCollection* (see appendix B3) was created. The *ReactionCollection* holds an array of reactions to loop through and call their *React()* method. The *ConditionCollection* holds one *ReactionCollection* and an array of conditions that when all satisfied (checked by *CheckAndReact()*) leads to the *ReactionCollection* calling its *React()* method. As conditions and reactions are *ScriptableObjects* they are not editable in the Unity inspector as well as not shown on the hierarchy.

To supervise all game mechanics multiple modular manager classes were created to govern their respective game mechanic, i.e. questManager for managing quest states.

Movement

Player movement and interaction was executed by mouse click. Therefore, an Event System component was added to an EventSystemManager gameobject to make it possible to send, receive and manage events evoked by a Pointer Click Event Trigger. All objects that intended to be motionless in the scene were marked as "Static", so when baking a NavMesh (a traversable area) the simulation knew that all objects that were marked as "Static" were non-traversable objects. A NavMeshAgent component was

added the player gameobject to identify that the player was an agent that should be able to traverse on the baked NavMesh area.

Player Movement

The *PlayerMovement* class was responsible for player movement and manipulating properties of the NavMeshAgent. More specifically the speed of the NavMeshAgent was slowed as it approached its target destination, and completely stopped when it reached its target destination. The speed of the NavMeshAgent's movement was controlled by a blend tree where thresholds for *idle*, *walk*, and *run* were specified (see appendix B4), based on an animator parameter, *speed*. The animator parameter *speed* was then manipulated by methods called in the *Update()* if the NavMeshAgent was in range of stopping or slowing (see figure 21).

```
private void Update()
{

    if (photonView.isMine)
    {

        if (agent.pathPending)
        {
            return;
        }

        float speed = agent.desiredVelocity.magnitude;

        if (agent.remainingDistance <= agent.stoppingDistance * stopDistanceProportion)
        {
            Stopping(out speed);
        }

        else if (agent.remainingDistance <= agent.stoppingDistance)
        {
            Slowing(out speed, agent.remainingDistance);
        }

        else if (speed > turnSpeedThreshold)
        {
            Moving();
        }

        animator.SetFloat(hashSpeedPara, speed, speedDampTime, Time.deltaTime);

    }
    else
    {
        syncTime += Time.deltaTime;
        transform.position = Vector3.Lerp(syncStartPosition, syncEndPosition, syncTime / syncDelay);
        transform.rotation = Quaternion.Lerp(syncStartRotation, syncEndRotation, syncTime / syncDelay);

        animator.SetBool("TakeObject", syncTakeObject);
        animator.SetFloat("Speed", syncSpeed);
    }
}
```

Figure 21 - Manipulating the animator parameter speed in the Update().

The *Update()* in *PlayerMovement* made use of a multiple IF statement. Firstly, we checked to see if the NavMeshAgent was currently calculating a path, if the result returned *True*, no action was taken as the NavMeshAgent was calculating a path; else *speed* was set equal to the speed that the NavMeshAgent was travelling at. Then another IF statement was used to see if the remaining distance on the NavMeshAgent's path was less than or equal to 10% of the total stopping distance, if the result returned *True*, then the *Stopping()* was called. Another IF statement was used to see if the remaining distance on the NavMeshAgent's path was less than or equal to the stopping distance, if the result returned *True*, then the *Stopping()* was called. One last IF statement was used to see if *speed* was greater than a quarter of the speed that the NavMeshAgent could travel at, if the result returned *True*, then the *Moving()* was called (which handles player rotation, not movement).

Regardless of the result of any IF statement the *speed* animator parameter was set equal to the speed variable in *PlayerMovement*, whilst introducing a small update delay of 0.1 seconds.

Stopping()

Stopping() (see appendix B5) firstly stopped the NavMeshAgent along its current path and set the transform position of the NavMeshAgent equal to the *destinationPosition*, and *speed* = 0. Then an IF statement was used to see if *currentInteractable* was not equal to null, if the result returned *True,* then the rotation of the NavMeshAgent was set to match the rotation on the *Interactable*, calling the *Interact()* in the *Interactable* script, and then setting *currentInteractable* to null to ensure that the *Interact()* was only called once.

Slowing()

Slowing() (see appendix B6) firstly stopped the NavMeshAgent along its current path then got how close the NavMeshAgent was to the destination/stopping distance, by dividing the remaining distance from the destination by the NavMeshAgent's stopping distance. A *proportionalDistance* = 1 depicted that the NavMeshAgent was on the destination, whereas a *proportionalDistance* = 0 conveyed that the NavMeshAgent was on the outer radius of the stopping distance. Then we checked if *currentInteractable* was not equal to null, if the result returned *True* then *targetRotation* was assigned the rotation of the *currentInteractable*, else if the result was *False* then *targetRotation* was assigned the rotation of the NavMeshAgent. The rotation of the NavMeshAgent was then interpolated from the NavMeshAgent's current position to the specified *targetRotation* based on the *proportionalDistance*. The position of the NavMeshAgent was then moved from the NavMeshAgent's current position to the specified *destinationPosition* over the timeframe *slowingSpeed*. Lastly, *speed* was set equal to the interpolation between slowingSpeed to 0 based on the *proportionalDistance*.

Moving()

Moving() (see appendix B7) set *targetRotation* equal to the rotation that the NavMeshAgent wanted to, and then set the rotation of the NavMeshAgent to an interpolation between the NavMeshAgent's current rotation and the *targetRotation* over the specified time frame.

OnGroundClick()

```
public void OnGroundClick(BaseEventData data)
{
    currentInteractable = null;

    PointerEventData pData = (PointerEventData)data;
    NavMeshHit hit;
    if (NavMesh.SamplePosition(pData.pointerCurrentRaycast.worldPosition, out hit, navMeshSampleDistance, NavMesh.AllAreas))
        destinationPosition = hit.position;
    else
        destinationPosition = pData.pointerCurrentRaycast.worldPosition;
    agent.SetDestination(destinationPosition);
    agent.Resume();
}
```

Figure 22 - The OnGroundClick() method.

The *OnGroundClick()* was executed when a Pointer Click event was recorded on the baked NavMesh, the interactable area was defined by a terrain collider component on the terrain gameobject. When executed, the BaseEventData obtained from an Event Trigger was casted to PointerEventData, and information regarding where the NavMesh was clicked by a PointerClick event was stored in *hit*. Then an IF statement was used to check if an area of the NavMesh was hit, if the result returned *True*, then *destinationPosition* was set to the position of the NavMesh that was hit. Else if the result returned False, then *destinationPosition* was set to the location that was obtained from the PointerEvent. The NavMeshAgent was then paused and its new path was calculated based on the given updated destination *destinationPosition*, after which the NavMeshAgent resumed along its current path (see figure 22).

OnInteractableClick()

```
public void OnInteractableClick(Interactable interactable)
{
    currentInteractable = interactable;
    destinationPosition = currentInteractable.interactionLocation.position;
    agent.SetDestination(destinationPosition);
    agent.Resume();
}
```

Figure 23 - The OnInteractableClick() method.

The *OnInteractableClick()* was executed when a Pointer Click event was recorded on an interactable object in the scene, the interactable area was defined by a box collider component on the interactable gameobject. When executed, the *Interactable* script that was attached to object in the scene was passed into the *OnInteractableClick()* as a parameter. *currentInteractable* was then set to the passed in *Interactable* script. *destinationPosition* was then set to the transform location of the interactable *interactionLocation* specified in the *Interactable* script. The NavMeshAgent was then paused and its new path was calculated based on the given updated destination *destinationPosition*, after which the NavMeshAgent was then resumed along its current path (see figure 23).

Interaction

As part of the game, the player must be able to interact with both non-playing characters (NPCs) and pick up items. For this we made use of an *Interactable* script (see appendix B8) to run reactions based on conditions. This *Interactable* script was called when the player moved close enough to the clicked object, and the movement entered the stopping radius (*Stopping()*). The *Interactable* script held a *ReactionCollection* and *ConditionCollection* used to check if an interaction should result in a set of reactions. On interaction, the script checked the conditions required to run reactions with the *CheckAndReact()* method (see appendix B2), if the conditions were not met, a default reaction was run which then specified an interaction is the type of reaction that was attached to its *ReactionCollection*.

For interactions with pick up items, we made use of *AnimationReaction, ConditionReaction, PickedUpItemReaction, SetActiveGameObjectReaction* and

ChangeQuestStateReaction (presented later). For interactions with NPCs, we used *ConditionReaction, ChangeQuestStateReaction* and *TextReaction*. The default reactions for interaction were *TextReactions*, which explained to the player that the currently attempted interaction was not valid at the given game state. *AnimationReaction* triggered an animation in the players *Animator* based on a string referencing the animation to play. *ConditionCollection* called an RPC (remote procedural call) method *ChangeCondition()* which took a Boolean and string input parameter and searched for a match in the conditions based on a hash value of that string, then setting the *satisfied* field to the Boolean input parameter. *PickedUpItemReaction* also made use of an RPC call, running *PickUpItem()*. This method took a string as input which was used to find the correct *Item* prefab in the resources folder. This item was then added to the inventory. Lastly, *SetActiveGameObjectReaction* used an RPC call *SetActive()*. *SetActive()* took a Boolean input which was the state to set the game object in. This game object was referenced in *SetActive()'s* script, as it was attached to the interactable object.

A simple inventory structure was used as there was only a need to show a sprite in the UI if an item had been picked up. To achieve this, the *Item* script held a sprite and was attached to a prefab (to be found in the resources folder). When an item was interacted with and added to the inventory, it was done by locating an open slot in the array of *Item* in *Inventory* (see appendix B9). An item was added to the inventory by setting a UI sprite to the item's sprite and enabling it.

User Interface

UIManager

The *UIManager* class was responsible for ensuring that the quest log was updated to display the current quest name and task description, and clearing the quest log when a quest is handed in (*Finished* state). The *SetupUIInformation()* was called in the *Start()* which displayed the value of the *questName* and *questDescription* property of the first quest in the *quests* list (see appendix B10). *UpdateUIInformation()* was called in the ChangeQuestState() and displayed the value of the *questName* and *questDescription* property of the current quest in the *quests* list (see appendix B11). *ClearQuestUIInformation()* was called in the ChangeQuestState() and updated the quest log to inform the player to the find the next quest (see appendix B12).

UpdateQuestIcon

The *UpdateQuestIcon* class was responsible for updating the diegetic quest icon user interface, *UpdateQuestIcon* ensured that the correct quest icon was displayed at the correct location for the current quest. An *Available* quest was displayed with a yellow exclamation mark, an *InProgress* quest was displayed with an empty exclamation mark, and a *Completed* quest was displayed with a yellow question mark.

Updating Quest Icons

UpdateQuestMarker() (see appendix B13), *UpdateBlackboardMarker()* (see appendix B14), and *UpdateHouseQuestMarker()* (see appendix B15) used a very similar methodology for updating quest icons. *UpdateQuestMarker()* utilised five input parameters: *QuestDest*, the NPC the quest was completed at, *QuestOrigin*, the NPC the quest was picked up from, and references to three different sprites: Town Mayor sprite, Town Mayor Assistant sprite and Farmer sprite. The *UpdateQuestMarker()* was called in the Update() three times with the following input parameters (see appendix B16), the description below describes the *UpdateQuestMarker()* using the first call for the method.

A multiple IF statement was used in *UpdateQuestMarker()*. First, an IF statement was used to check if the *quests* list was empty, if the result returned *True*, then all sprite renderer components on all NPCs were set to null. Then another IF statement was used to check if the current quests questState was *NotAvailable*, if the result returned *True*, then the sprite renderer components on the Town Mayor Assistant was set to null. Another IF statement was used to check if the current quests questState was *Available*, and whether the current quest can be picked up from the Town Mayor Assistant, if the result returned *True*, then the sprite renderer component on the Town Mayor Assistant was set to the *filledExclamationMark* sprite. Another IF statement was used to check if the current quests questState was *InProgress*, and whether the current quest can be picked up from the Town Mayor Assistant, if the result returned *True*, then the sprite renderer component on the Town Mayor Assistant was set to the *emptyExclamationMark* sprite. Then another IF statement was used to check if the current quests questState was *Completed*, and whether the current quest should be handed into the Town Mayor Assistant, if the result returned *True*, then the sprite renderer component on the Town Mayor Assistant was set to the *filledQuestionMark* sprite, and the sprite renderer component on the Town Mayor and Farmer is set to null.

TextManager

The *TextManager* class was responsible for ensuring that UI text was displayed with the specified message, colour and delay when a *TextReaction* was executed in a *ReactionCollection*. The *DisplayManager()* in the *TextManager* class was called by the *ImmediateReaction()* in the *TextReaction* class (see appendix B17) using an RPC call, with specified input parameters. The *Update()* used an IF statement to check to see if an *Instruction* existed in the *instructions* list, and whether the time elapsed was greater than the *startTime* specified in the Instruction, if the result returned *True*, then the message was displayed. Else IF the time elapsed was greater than the specified *clearTime* then the message was cleared.

DisplayMessage()

DisplayMessage() (see appendix B18) first assigned the colour of the text *textColor* to the passed in R, G, B values, then *startTime* for the message to display was established by specifying the passed in *delay* parameter. A *displayDuraton* was calculated by getting the number of characters in the message and multiplying each character by 0.1 seconds, whilst adding an additional 0.5 seconds to the *displayDuration*. Then *newClearTime*

stored the value of adding the *displayDuration* and the previously specified delay time *startTime*. Then an IF statement was used to check if *newClearTime* was greater than *clearTime*, if the result returned *True*, then the value of *newClearTime* was assigned to *clearTime*. A new Instruction was then made with the following prespecified variables: *speaker, message, textColor, startTime*. The new *Instruction* was then added to the *instructions* list, then calling the *SortInstructions()*, which was responsible for organising and displaying *Instructions* with a shorter *startTime* first.

AnswerButton

The *AnswerButton* script was attached to each answer button on each blackboard question UI. *AnswerButton* has the *SelectAnswer()* which called the *SelectAnswerRPC()* RPC. *SelectAnswer()* was called when an *OnClick()* event was recorded on the button and passed an integer value to the *SelectAnswer()*.

SelectAnswerRPC()

SelectAnswerRPC() (see appendix B19) passed in an integer value specified in the editor (A = 0, B = 1, C = 2, D = 3). *selectedAnswer* was then set to the integer value specified on the button that was clicked. Then the Outline component was enabled on the button that was clicked, and disabled on the rest.

SubmitButton

When the submit button was clicked, the *SubmitButton* class was responsible for submitting the answer of the currently selected button, disabling the blackboard UI and question UI, and clearing the *questSpeaker* and *questDialogue* text fields. Whilst additionally ensuring that players could only submit an answer for a *QuestionQuest* once.

SubmitAnswerRPC()

```
[PunRPC]
public void SubmitAnswerRPC()
{
    QuestionQuest qq = (QuestionQuest)questManager.quests[0];
    qq.CheckAnswer(uiManager.selectedAnswer);

    foreach (Condition c in AllConditions.Instance.conditions)
    {
        if (c.hash == Animator.StringToHash(conditionName))
        {
            Debug.Log(c.description);
            c.satisfied = false;
        }
        if (c.hash == Animator.StringToHash(conditionNameBecomingTrue))
        {
            c.satisfied = true;
        }
    }
    questManager.quests[0].questState = QuestState.Completed;

    questionPart.SetActive(false);
    blackboard.SetActive(false);

    // Clear these fields when the blackboard is closed by the player.
    uiManager.questSpeaker.text = "";
    uiManager.questDialogue.text = "";
}
```

Figure 24 - The SubmitAnswer() RPC method.

SubmitAnswerRPC() (see figure 24) was called using an RPC call in the *SubmitAnswer()*, and ensured that the *SubmitAnswerRPC()* could only be executed if an answer had been selected prior to pressing the submit button.

First, the current *QuestionQuest* was assigned to *qq*, and then the *CheckAnswer()* was called while passing in the integer value of the player's selected answer. We looped through all *Conditions* in the *AllConditions* asset, then an IF statement was used to check if the hash of the current condition matched the generated hash of the *conditionName* specified in the editor on the submit button, if the result returned *True*, then the satisfied field on the *Condition* in the *AllConditions* asset was set to *False*. Then another IF statement was used to check if the hash of the current condition matched the generated hash of the *conditionNameBecomingTrue* specified in the editor on the submit button, if the result returned *True*, then the satisfied field on the *Condition* in the *AllConditions* asset was set to *True*. After selecting an answer and pressing the submit button, the *questState* property for the current *QuestionQuest* was set to *Completed*, whilst simultaneously disabling the blackboard and question UI, and clearing the *questSpeaker* and *questDialogue* UI text fields if the delay still had not expired.

ShowUIRPC

The *ShowUI()* (see appendix B20) in *ShowUIRPC* was called from *ShowUIElementReaction*, which simply enables the blackboard UI and specified question UI.

Quests

An *Interactable* script component was placed on each NPC gameobject, each *Interactable* script held a transform location of the NPC, a *ConditionCollection* (a collection of conditions), and a reference to a gameobject that held a *ReactionCollection* (a collection of reactions). All game events in the *ReactionCollection* were executed when all conditions in the *ConditionCollection* were satisfied, e.g. if the quest "The First Job Part 1" had been completed, then execute the referenced *ReactionCollection* "FinishingTheFirstJobPart1Reaction" in the "FinishingTheFirstJobPart1" *Condition*, (see appendix B21). The *ReactionCollection* then executed all Reactions in the *ReactionCollection*: *ChangeQuestStateReaction*, changed the *questState* of the specified quest. *ConditionReaction*, changed the Boolean *Satisfied* state of the specified condition (see appendix B22).

Quest Class Structure and Inheritance

The quest infrastructure derives from field instances and a *Quest* constructor in the *Quest* abstract class. Three different *Quest* subclasses (LocateQuest, QuestionQuest, CollectQuest) then inherited common properties of a quest from the Quest abstract class: *questName*, name of the quest. *questDescription*, a description of the goal of the quest. *questScore*, the number of gold stars obtained from the quest. *hashID*, the name of the quest converted to a hashID. *questState*, the state of the quest, i.e., *InProgress* or *Available*. questOrigin, where the quest was obtained. *questDestination*, where the quest was handed in; whilst defining further exclusive properties for each quest type in their

relevant sub-class's constructor, i.e. a string property to store the question text for a *QuestionQuest* quest.

Enums as Properties

The *Quest* abstract class and *QuestionQuest*/*LocateQuest*/*CollectQuest* subclasses of the abstract class used enum properties (see appendix B23). By doing so it made it easier to select values from an enum list in the editor, rather than hard coding a value in the editor.

QuestManager

The *QuestManager* class was responsible for instantiating all game quests, separating quests from finished quests, and updating the quest log by calling methods in the *UIManager* class to display the correct information regarding the current quest.

InstantiateQuests()

```
public void InstantiateQuests()
{
    // Questline 1 - The Initiation.

    // 1.1 Locate Quest - Find and talk to the assistant.
    Quest L1quest1 = new LocateQuest("The Initiation Part 1", "Find og tal med borgmesterens assistent.",
        0, QuestState.Completed, QuestOrigin.GameStart, QuestDest.TownMayorAssistant);
    quests.Add(L1quest1);

    // 1.2 Question Quest - What is the probability of rolling a six on a six-sided die?
    Quest Q1quest1 = new QuestionQuest("What is the probability of rolling a six on a six-sided die?",
        new string[] { "A: 1 in 6", "B: 1 in 4", "C: 1 in 3", "D: 1 in 2" }, 0,
        "The Initiation Part 2", "Besvar borgmesterens assistents spørgsmål (1/1)", 0,
        QuestState.NotAvailable, QuestOrigin.TownMayorAssistant, QuestDest.TownMayorAssistant);
    quests.Add(Q1quest1);
```

Figure 25 - A small section of the InstantiateQuests() method.

Being called in Unity's *Awake()* method, *InstantiateQuests()* simultaneously instantiated all quests and defined all properties of each *Quest* subclass (QuestionQuest, LocateQuest, CollectQuest) that were inherited from the *Quest* Abstract class, and properties that were exclusive to each subclass (see figure 25). All quests were instantiated in *Not Available* state on game start (excluding the first quest, as the quest was automatically assigned to the player on game start).

ChangeQuestState()

```
[PunRPC]
public void ChangeQuestState(QuestState questState, string questName)
{
    int hash = Animator.StringToHash(questName);

    for (int i = 0; i < quests.Count; i++)
    {
        if (quests[i].hashID == hash)
        {
            quests[i].questState = questState;

            if (quests[i].questState == QuestState.InProgress || quests[i].questState == QuestState.Completed)
            {
                uiManager.UpdateUIInformation();
                return;
            }

            if (quests[i].questState == QuestState.Finished)
            {
                MoveToFinishedList(i);
                uiManager.ClearQuestUIInformation();
                uiManager.scoreManager.CalculateCurrentTotalGoldStars();
            }
            return;
        }
    }
}
```

Figure 26 - The ChangeQuestState() method.

The *ChangeQuestState()* was tagged with a *[PunRPC]* tag (see figure 26), thus allowing the method to utilise PUN's remote procedure call (to broadcast a method to all clients in the same room). The *ChangeQuestState()* was called by the *ChangeQuestStateReaction()* (which inherits from the abstract class *Reaction*) using an RPC call, where the parameters *questState* and *questName* were defined in the *ChangeQuestStateReaction* Reaction in a *ReactionCollection* in the Unity editor. *ChangeQuestState()* then generated a hashID from the *questName* string specified in the *ChangeQuestStateReaction* Reaction in the editor, and looped X number of times based on the number of quests contained in the *quests* list. Then an IF statement was used to check whether the generated hashID matched the hashID of the current quest in the loop, if the result returned *True* then *questState* was set to the quest state value specified in the *ChangeQuestStateReaction* Reaction in the editor. Then another IF statement was used to check whether the *questState* of the current quest in the loop was equal to *InProgress* or *Completed*, if the result returned *True* then the *UpdateUIInformation()* was called in the *UIManager* class. Else IF the *questState* of the current quest in the loop was equal to *Finished*, then the *MoveToFinishedList()* was called, using the current quest in the loop as an input parameter for the method. Furthermore, the *ClearQuestUIInformation()* and *CalculateCurrentTotalGoldStars()* was called in the *UIManager* and *ScoreManager* class, respectively.

MoveToFinishedList()

```
public void MoveToFinishedList(int i)
{
    if (quests[0].questState == QuestState.Finished && !finishedQuests.Contains(quests[0]))
    {
        finishedQuests.Add(quests[i]);

        if (quests.Contains(quests[i]))
        {
            quests.Remove(quests[i]);
        }
    }
}
```

Figure 27 - The MoveToFinishedList() method.

MoveToFinishedList() sorted quests into two lists: *finishedQuests*, quests in *Finished* state. *quests*, all other quests (see figure 27). An IF statement was used to check whether the current quest was in *Finished* state and that it was not already contained in the *finishedQuests* list, if the result returned *True* then the current quest was added the *finishedQuests* list. Afterwards, another IF statement was used to check whether the current quest was still contained in the *quests* list, if the result returned *True* then the current quest was removed from the *quests* list.

Networking

Networking was handled using PUN, a networking Unity asset, enabling quick and efficient networking using cloud servers. Using the PUN API in a *NetworkManager* script, the game ensured that players join a lobby and room when connecting to a server. For testing purposes the auto-join feature was used as only one group could play at a time; although player count restrictions were enabled on the room. Furthermore, by using the PUN API it can be assured that when joining or failing to join a room, the player creates a new room to play in. When joining a room, the game checked whether the player count of the current room exceeded the prespecified threshold before instantiating a player.

Synchronization and Serialization

Data sharing was implemented by using PUN's serialization techniques; RPCs and *PhotonViews*. PhotonViews were attached to every instantiated object that needed to be synchronized between players, making it possible for the network to recognize local objects. Attaching a PhotonView to a game object allowed for RPC calls (found in a script on the game object) in the reactions. To recognize methods as available for RPC calls, methods were given the PunRPC attribute, making it recognisable by PUN's PhotonView.RPC() method; used to call methods remotely. The PhotonView was used in serializing player movement and animation data using PUN APIs *OnPhotonSerializeView()* method placed in the *PlayerMovement* script. In *OnPhotonSerializeView()* a PhotonStream wrote data to be read by receivers. Data regarding movement was the player's transform, rotation and velocity, and for animation, a Boolean and float for item pick up and walking. When reading from the stream, the data was cast from objects to their respective types for them to be saved locally and used on the game object with the correct PhotonView. To take into

consideration the delay of sending data during gameplay, the synchronized velocity was multiplied by a *syncDelay* and added to the position, smoothening the visual changes through prediction; The rotation was also smoothed by multiplying it by syncDelay.

In the *Update()* these locally saved values were used to change the respective properties of the objects which were not the player's PhotonView. There we make use of Vector3.Lerp() and Quaternion.Lerp() for altering position and rotation, and Animator.SetBool() and Animator.SetFloat() for setting values in the player's animator controller. The Lerp method interpolated values from a start position and rotation to an end value being the values read from the stream (see figure 28).

```
else
{
    syncTime += Time.deltaTime;
    transform.position = Vector3.Lerp(syncStartPosition, syncEndPosition, syncTime / syncDelay);
    transform.rotation = Quaternion.Lerp(syncStartRotation, syncEndRotation, syncTime / syncDelay);

    animator.SetBool("TakeObject", syncTakeObject);
    animator.SetFloat("Speed", syncSpeed);
}
```

Figure 28 - Synchronization update when PhotonView is not owned by the player.

In the case of the PhotonView being owned by the current player, the player's own movement was processed (see appendix B25).

EventTriggers

On game start, when a player joins the room, all fields on all *EventTriggers* were populated.

```
public void CreateEventTrigger()
{
    if (NetworkManager.PlayerCount == 1)
        player = GameObject.Find("Player 1(Clone)");

    if (NetworkManager.PlayerCount == 2)
        player = GameObject.Find("Player 2(Clone)");

    EventTrigger trigger = gameObject.GetComponent<EventTrigger>();
    EventTrigger.Entry entry = new EventTrigger.Entry();
    trigger.triggers.Add(entry);
    entry.eventID = EventTriggerType.PointerClick;
    entry.callback.AddListener((eventData) => {
            player.GetComponent<PlayerMovement>().OnInteractableClick(gameObject.GetComponent<Interactable>()); });
}
```

Figure 29 - Setting up EventTriggers for interactable objects.

First, references were set to a player controlled game object based on the player count in the room. Second, the *EventTrigger* was given a new entry of the *PointerClick* type so it would trigger on mouse click. Lastly, a listener was added to the entry, calling either the OnInteractcableClick() or OnGroundClick() methods for the respective player game object. These scripts which initialised the EventTriggers were attached to all interactable objects and NavMeshes, to hold a reference to itself allowing for the listener call to use the correct input parameter.

Data Management

GameStartUp

The *GameStartUp* class was responsible for initializing and stopping a timer which was used to measure the time of the game session (see appendix B26), and resetting the global state of all (Boolean) *satisfied* fields of all conditions saved in the *AllConditions* asset back to *False* on game start, except *gameStart* (see appendix B27). When one quest existed in the *finishedQuest* list, the timer was instantiated. The timer was created in the *Update()*, and therefore the *startTimer* Boolean was used to ensure that the timer was only instantiated once. The reason that the timer only started when one quest had been added to the *finishedQuest* list was to give us an opportunity to prepare the test environment for the next test participant, without skewing the timer data. When no quests existed in the *quests* list, which implies that all quests have been completed, the timer was stopped. A *wait* Boolean was used to ensure that the timer was only stopped once, as it was called in the *Update()*.

ScoreManager

The *ScoreManager* class was responsible for calculating the following information:

- Total stars
- Current total stars
- Stars earned for each quest line
- Mega stars earned for each quest line
- Total mega stars.

The subsequent information was displayed in the final score screen at the end of the game; whilst the current total stars was also displayed in the game UI.

CalculateCurrentTotalGoldStars()

CalculateCurrentTotalGoldStars() calculated the current total stars earned by the player and was called every time a quest was Finished in the *ChangeQuestState()*. *CalculateCurrentTotalGoldStars()* used an IF statement to check whether the quest that was just added to the *finishedQuest* list was of the type *QuestionQuest* (stars are only awarded for correct answers in a QuestionQuest), if the result returned *True* then the quest was assigned to the variable *q*. Another IF statement was used to check whether the *CheckAnswer()* returned *True*, using the integer value of the player's selected answer *selectedAnswer* as a parameter. The *CheckAnswer()* assigned the value of *selectedAnswer* to *playerAnswerIndex*, and then checked to see if the integer value of *playerAnswerIndex* matched the *QuestionQuest*'s correct answer index *correctAnswerIndex*. If *CheckAnswer()* returned *True*, then *totalStars* was incremented by 1, the *questScore* property of the *QuestionQuest* was set = 1, and the text in the UI was updated with the new *totalStars* value. Regardless of whether the answer was correct or incorrect, *selectedAnswer* was set to -1 afterwards, to ensure answers were not highlighted and that an answer must be selected for the next question (see appendix B28).

EndGameCalculateTotalGoldStars()

EndGameCalculateTotalGoldStars() was called in the *CalculateMegaStars(),* and calculated and displayed the total stars earned by the player. The total stars were calculated by looping through the list of all finished quests in the *finishedQuests* list. The *questScore* property was obtained for each looped quest and stored temporarily in *subTotalStars* which then was added to the total *endTotalStars*. *endTotalStars* was then converted from an integer to a string and displayed in the final score screen (see appendix B29).

CalculateMegaStars()

CalculateMegaStars() was executed when all quests were *Finished* and contained in the *finishedQuests* list (see appendix B30). This was due to the fact that *CalculateMegaStars()* was only responsible for calling calculation related methods and performing calculations for scores that were to be displayed on the final score screen. *CalculateMegaStars()* calculated stars earned for each quest line by adding the *questScore* property of relevant indexed quests for each quest line. The total stars for each quest line were then displayed in the final score screen by calling *DisplayGoldStarResults()* (see appendix B31). Additionally, the *CalculateMegaStars()* called the following methods which calculated total mega stars *totalMegaStarsResult* and questline specific stars: *CalculateTheInitiationMegaStars(), CalculatePreparationIsKey(), VisitingTheFarmer()* Lastly, *CalculateMegaStars()* converted the integer value of *totalMegaStarsResult* to a string and displayed it in the final score screen.

Calculating MegaStars and Questline Specific MegaStars

The *CalculateTheInitiationMegaStars()* (see appendix B32), *CalculatePreparationIsKey()* (see appendix B33) and *VisitingTheFarmer()* (see appendix B34) used the same methodology for calculating mega stars and quest line specific mega stars. The *CalculateTheInitiationMegaStars()* will be explained as an example.

The total earned questline specific stars for each quest line were obtained from the *CalculateMegaStars()* mentioned previously. An IF statement with multiple *theInitiationTotalStars* value thresholds were used to ensure the correct calculation and display operations were performed, based on the value of *theInitiationTotalStars*.

The following operations were performed in each IF condition:

1. Setting a temporary integer variable to the number of mega stars earned, *totaltheInitiationMegaStars* and adding the value of *totaltheInitiationMegaStars* to the current total of mega stars earned, *totalMegaStars*.
2. Enabling the correct gameobject that holds an image component displaying the correct number of quest line specific mega stars earned.

SaveDataToTxtFile

The *SaveDataToTxtFile* class was responsible for writing the following data to a .txt file:

- Score for each finished quest.
- Player's answer for each QuestionQuest.

- Mega stars earned for each of quest chain.
- Total mega stars earned.
- Total gold stars earned.
- Time of game session.

The *SaveData()* which was responsible for writing the .txt file was executed when the application was closed via the *OnApplicationQuit()*. The filename of the .txt file was titled by the current system time and type of test condition (Condition HH_MM_SS), e.g. *"Coop 14_33_40"*.

SaveData()

```
public void SaveData()
{
    string[] lines = new string[26];
    string[] liness = new string[6];
    int i = 0;

    foreach (Quest q in questManager.finishedQuests)
    {
        if (q.GetType().Equals(typeof(QuestionQuest)))
        {
            QuestionQuest qq = (QuestionQuest)q;
            lines[i] = "Name: " + qq.questName + ", Type: " + qq.GetType() + ", Score: " + qq.questScore +
                ", Player Answer: " + qq.playerAnswerIndex + ", Correct Answer: " + qq.correctAnswerIndex;
        }
        else
        {
            lines[i] = "Name: " + q.questName + ", Type: " + q.GetType() + ", Score: " + q.questScore;
        }
        i++;
    }
    liness[0] = "The Initiation Mega Stars Earned: " + ScoreManager.totaltheInitiationMegaStars.ToString();
    liness[1] = "Preparation Is Key Mega Stars Earned: " + ScoreManager.totalpreparationIsKeyMegaStars.ToString();
    liness[2] = "Visiting The Farmer Stars Earned: " + ScoreManager.totalvisitingTheFarmerMegaStars.ToString();
    liness[3] = "Total Mega Stars Earned: " + ScoreManager.totalMegaStars.ToString();
    liness[4] = "Total Stars Earned: " + ScoreManager.endTotalStars.ToString();
    liness[5] = "Time of Game Session: " + GameStartSetup.timer.Elapsed.ToString();

    var c = lines.Concat(liness).ToArray();
    string dateTimestamp = System.DateTime.Now.Hour + "_" + System.DateTime.Now.Minute + "_" + System.DateTime.Now.Second;
    string gameType = "";

    if (NetworkManager.PlayerCount > 1)
    {
        gameType = "Coop";
    }
    else
    {
        gameType = "Solo";
    }

    File.WriteAllLines(gameType + " " + dateTimestamp + ".txt", c);
}
```

Figure 30 - The SaveData() method.

The *SaveData()* looped through all quests in the *finishedQuests* list (see figure 30). Then an IF statement was used to check whether the quest was of the type *QuestionQuest* (a *QuestionQuest* requires different properties to be labelled), if the result returned *True* then the properties of the *QuestionQuest* object were saved and labelled in a string on the correlating index of the *lines* string array. Else if the result returned *False* then the properties of the quest (which is not a *QuestionQuest*) were also saved and labelled in a string on the correlating index of the *lines* string array. In a new string array *liness*, the following data that was mentioned earlier is saved in independent indexes.

Both the *lines* and *liness* string array were concatenated to a single array *c*, then the current system time was obtained and stored in *dateTimestamp*, and the *gameType* was obtained by checking how many players were connected to the room. The file was then created and written using the concatenated array in the Unity project folder.

Editors

Reaction and Condition Editors

As mentioned, editors were used to allow easy changes and set-up of conditions and reactions, based on the Unity tutorial. For this an inheritence structure was used, having abstract classes (*ReactionEditor* and *ConditionEditor*) hold the needed information general for all editors to be made for condition- and reactions scripts. The *ReactionEditor* drew the reactions on the inspector GUI using Unity's *Editor's Foldout* making a label for the created reaction. In concrete implementations of reaction editors, e.g. *ChangeQuestStateReactionEditor*, there were two options for setting up what to show in the inspector overriding the *DrawReaction()* and *GetFoldoutLabel()*. In both cases the editor script inherited from the *ReactionEditor*. The first implementation option was making use of *ReactionEditor's* default drawing, and there is only a need to override the *GetFoldoutLabel()* where in a string is returned and used to set the name of the reaction item in the inspector. The second implementation option overrided both presented overridable methods, and was used when the default drawing was not preferred. Here the editor script was given *SerializedProperty* fields that was then set to represent the corresponding fields in the script the editor looked at. Overriding *GetFoldoutLabel()* worked as in the first implementation option, but *DrawReaction()* was overridden to one's preferences. This could for example be in the *ChangeQuestStateReactionEditor* where the text box for quest name was set to be drawn in a specific size. As the default drawing was overridden, the new version must also draw the rest of the *SerializeProperty* fields. For an editor script to know which script to look at and present properties, the editor script was given the *CustomEditor* attribute, which took an input of a type, being the inspected type (*CustomEditor(T)*).

The *ConditionEditor* script could draw the conditions in two ways in the inspector: for when they are part of the *AllConditions* or the *ConditionCollection*. If part of *AllConditions*, only the description field of the condition was drawn, together with a button for removing it from the list. If part of *ConditionCollection*, a popup was drawn, containing all the condition descriptions found in the *AllConditions* condition list. Selecting this popup set the description property of the condition being presented by the editor. Furthermore, a checkbox (Unity's representation of a Boolean) for the condition's *satisfied* field, and a button for removing the condition again was drawn. *SerializedObject.ApplyModifiedProperties()* was called in the end to ensure the changed properties in the editor is set in the actual condition object. The *AllCondtionsEditor* was responsible for creating editors for conditions created in the list. This editor held methods for creating and removing conditions; removing complexity from the *AllConditions* itself. To instantiate an *AllConditions* object, as it cannot be attached to a game object as a result of being a ScriptableObject, the method *CreateAllConditionsAsset()* allowed for instantiating from the menu.

Collection Editors

Editors were also made for the *ConditonCollection* and *ReactionCollection*. An abstract class, *EditorWithSubEditors<TEditor, TTarget>*, was used for taking care of creating editors for each subeditor. Extensions were made with *ConditionCollectionEditor* and *ReactionCollectionEditor*. *ConditionCollectionEditor* allowed the creation and removing of condition collections. Making use of *ExpandedGUI()* (which is called when the collection is folded out) the editor set up labels and called *OnInspectorGUI()* for all subeditors, the condition editors, which ensures they were drawn. *ReactionCollectionEditor* ensured editors were created for all reactions in the collection. As the editor for condition collections, *ReactionCollectionEditor* called each subeditor *OnInspectorGUI()*. It further drew a dropdown and button for selecting a type of reaction and adding it to the collection. *SetReactionNamesArray()* populated a *Type* array based on reactions which were used in the popup. Based on the selection index in the popup, the element on the index in the *Type* array was the type of the reaction to create.

User Study

Test Methods

Test participants were selected based on pre-specified demographic characteristics: educational level, as well as availability. Thus, the selected sampling strategy was non-probability sampling as some populations were deliberately hidden (as they did not fulfil the demographic characteristic criteria) and each member of the population under study did not have an equal chance of being selected to participate in the study. Since pre-specified characteristics of test participants were of great importance regarding the objective and overall success of the experiment, a combination of convenience and quota sampling were used.

The experiment consisted of 24 test participants (n = 24), all of whom were third grade students from Lindehøjskolen in Herlev. The age of all individuals was between the ages of 8 - 10 years old, 13 of which were male and the remaining 11 were female.

The data gathered was of quantitative and qualitative value. Qualitative data was gathered through questionnaires, observations and short mini-focus group interviews (consisting of 2 individuals) where only introductory type questions (Bjørner & Forlag, 2015) were asked.

Ordinal data was gathered from an Individual and Cooperation questionnaire using dichotomous and four-point Likert scale values. Nominal and Psychographic (Bjørner & Forlag, 2015) data was also obtained from the Individual and Cooperation questionnaire to establish each test participant's generic interest in computers and computer games; as someone with the same mathematical capability who uses a computer on a daily basis and frequently plays computer games is more likely to perform better than someone who does not. Lastly, ratio data in the form of time and number of correct answers was

recorded during each unique gameplay session and stored in a relevant variable which was then written to a text file for quantitative data analysis.

Test Environment & Structure

The experiment was conducted at Lindehøjskolen over the course of two days. Two rooms were reserved for the experiment, and a notice outside each room was put on display to ensure no unexpected entrances or disturbances occurred during testing. The experiment utilised three test conditions: individual DGBL (see figure 31), physical cooperative DGBL (see figure 32), and virtual cooperative DGBL (see figure 33). A staff room was used for all three test conditions, and a second room (a spare classroom) was used to support the virtual cooperative condition. Both rooms were approximately 36 square meters in size; to facilitate an optimal learning environment the windows were opened to ventilate the room, and the doors were shut closed to prevent distraction and ensure silence.

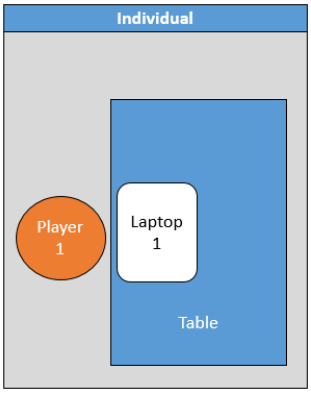


Figure 31 - The individual DGBL condition.

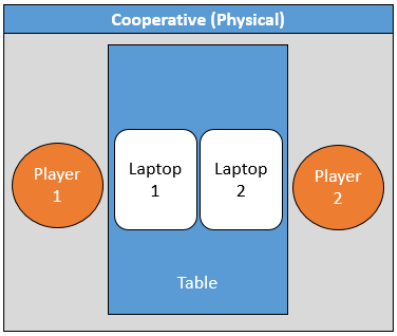


Figure 32 - The cooperative DGBL condition with physical cooperation medium: participants cooperated in the same physical space (room).

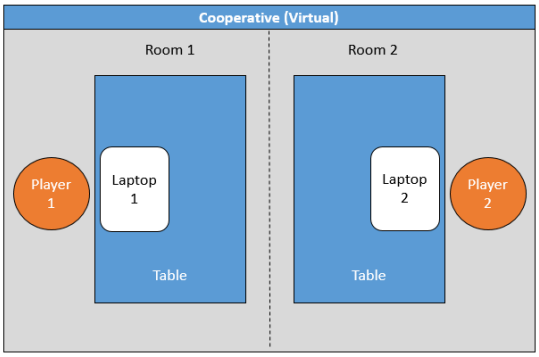


Figure 33 - The cooperative DGBL condition with virtual cooperation medium; participants cooperated using a virtual medium (Discord) while sitting in different rooms.

Testing Procedure

Initially, a pilot test was conducted on two individuals (1 male and 1 female, between the ages of 28 - 39) to ensure that as many bugs were identified as possible before composing the final project build, and to ensure that the English to Danish translation of the narrative was accurate. The goal of the experiment was to answer two research questions, specified in the problem statement:

Research Question 1

"Is a cooperative DGBL experience more effective than an individual DGBL experience?"

Research Question 2

"Is the process of cooperation more effective over a virtual medium than in a common physical space?"

All test subjects who participated in the physical and virtual cooperation medium condition for research question 2, were also considered to have participated in a cooperative DGBL experience in research question 1. To answer both research questions, the experiment was structured into a two-phase process, utilising the scientific control method in both cases; where the independent variable being altered was unique between the two research questions (see appendix C1).

Research Question 1

Independent Variable - Social Setting

Phase 1 - Individual GBL test/cooperative GBL test.

Phase 2 - Post-test individual/cooperative questionnaire.

Research Question 2

Independent Variable - Cooperative Medium

Phase 1 - Physical cooperative GBL test/virtual cooperative GBL test.

Phase 2 - Post-test cooperative questionnaire.

For research question 1 the treated group were introduced to a cooperative DGBL environment (whether it is physical or virtual), and the control group participated in an individual DGBL environment, where the following dependent variables were observed: *Motivation*, *Precision* and *Speed*. For research question 2 the treated group were introduced to a virtual cooperative DGBL environment and the control group engaged in a physical cooperative DGBL environment, where the following dependent variables were observed: *Cooperation*, *Precision* and *Speed*.

Results

Results used to answer the problem statement were obtained from the game, observations, interviews and questionnaires. Both quantitative and qualitative data was analysed and depicted using both descriptive and inferential statistics. Quantitative data was classified as: the completion speed of the game, and the total number of correct answers. Qualitative data consisted of questionnaire responses relating to motivation, cooperation and individualism-collectivism. To determine statistical significance of test data the following statistical tests were used: Wilcoxon rank sum test or two-sample t-test; based on the result of the Anderson-Darling test. Therefore, to investigate research question 1; hypothesis 1 was formulated, and to investigate research question 2; hypothesis 2 was formulated.

Hypothesis 1 - **Individual vs. Cooperative DGBL Experience:**

The null hypotheses for the three dependent variables were stated as the following:

1. An individual GBL environment is just as effective than a cooperative GBL environment.
    a. **Motivation** - The rank sum of motivation for the treated group and the control group is the same.
    b. **Precision** - The two-sample t-test of precision for the treated group and the control group is the same.
    c. **Speed** - The rank sum of speed for the treated group and the control group is the same.

Therefore, the alternative hypotheses for the three dependent variables were:

1. A cooperative GBL environment is more effective than an individual GBL environment.
    a. **Motivation** - The rank sum of motivation for the treated group is higher than the control group.
    b. **Precision** - The two-sample t-test of precision for the treated group is higher than the control group.
    c. **Speed** - The rank sum of speed for the treated group is higher than the control group.

Hypothesis 2 - **Physical vs. Virtual Medium:**

The null hypotheses for the three dependent variables were stated as the following:

1. The process of cooperation is just as effective when using a common physical space than a virtual medium.
    a. **Cooperation** - The two-sample t-test of cooperation for the treated group and the control group is the same.
    b. **Precision** - The two-sample t-test of precision for the treated group and the control group is the same.
    c. **Speed** - The two-sample t-test of speed for the treated group and the control group is the same.

Therefore, the alternative hypotheses for the three dependent variables were:

1. The process of cooperation is more effective when using a common physical space than a virtual medium.
    a. **Cooperation** - The two-sample t-test of cooperation for the treated group is higher than the control group.
    b. **Precision** - The two-sample t-test of precision for the treated group is higher than the control group.
    c. **Speed** - The two-sample t-test of speed for the treated group is higher than the control group.

Quantitative Results

Anderson-Darling normality tests were performed on six data sets; one data set for each dependent variable in both hypotheses. The data sets and results of their respective Anderson-Darling tests are as follows:

Hypothesis 1:

- Motivation in individual GBL environment vs. Motivation in individual GBL environment
  - **Result:** not normally distributed.
- Precision in individual GBL environment vs. Precision in cooperative GBL environment
  - **Result:** not normally distributed.
- Speed in individual GBL environment vs. Speed in cooperative GBL environment
  - **Result:** not normally distributed.

Hypothesis 2:

- Cooperation in physical cooperative GBL environment vs. Cooperation in virtual cooperative GBL environment
  - **Result:** normally distributed.
- Precision in physical cooperative GBL environment vs. Precision in virtual cooperative GBL environment
  - **Result:** normally distributed.
- Speed in physical cooperative GBL environment vs. Speed in virtual cooperative GBL environment
  - **Result:** normally distributed.

Important to note is the fact that of the total 24 test participants, 18 (10 physical cooperative medium & 8 virtual cooperative medium) participated in the cooperative DGBL condition and 6 participated in the individual DGBL condition. All six data sets are illustrated in box plots below (see figure 34, 35, 36, 37, 38, and 39).

Motivation - Individual DGBL vs. Cooperation DGBL

Comparing motivation in individual DGBL and cooperative DGBL resulted in a significance value of $p = 0.67$; thus, resulting in the acceptance of the null hypothesis for factor **a** of hypothesis 1.

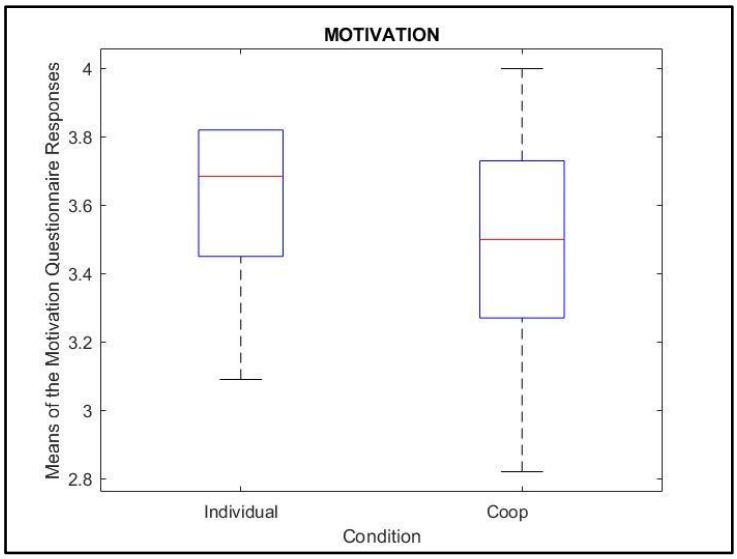


Figure 34 - Boxplot of means of questionnaire answers relating to motivation

As seen in figure 34, the individual DGBL data set possessed a higher median as well as a higher minimum, indicating more motivation in individual DGBL. However, the cooperative DGBL data set suggests that a higher level of motivation can be achieved in cooperative DGBL (due to a higher maximum).

Precision - Individual DGBL vs. Cooperation DGBL

Comparing precision in individual DGBL and cooperative DGBL resulted in a significance value of $p = 0.18$; thus, resulting in the acceptance of the null hypothesis for factor **b** of hypothesis 1.

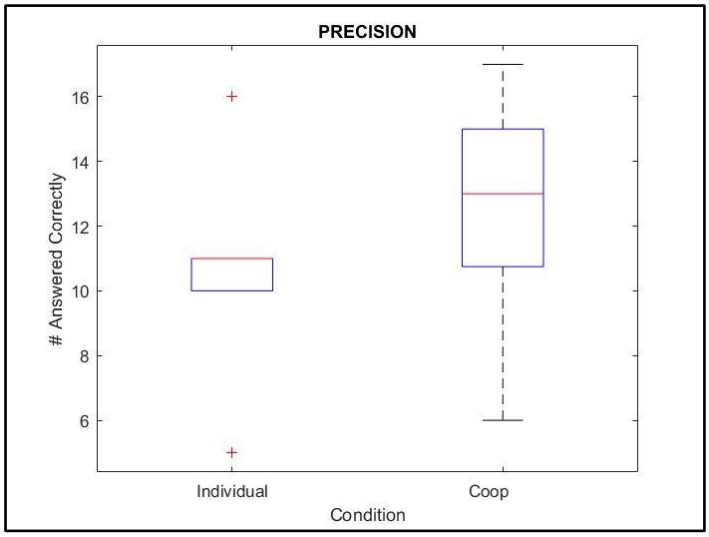


Figure 35 - Boxplot of correctly answered mathematics questions between the individual and cooperative conditions

Figure 35 indicates that students provided more correct answers in a cooperative DGBL environment; the cooperative condition also achieved a higher median and maximum value. Moreover, the boxplot for the individual DGBL condition conveys two data points that are outliers of the individual dataset.

Speed - Individual DGBL vs. Cooperation DGBL

Comparing speed in individual DGBL and cooperative DGBL resulted in a significance value of $p = 0.43$; thus, resulting in the acceptance of the null hypothesis for factor **c** of hypothesis 1.

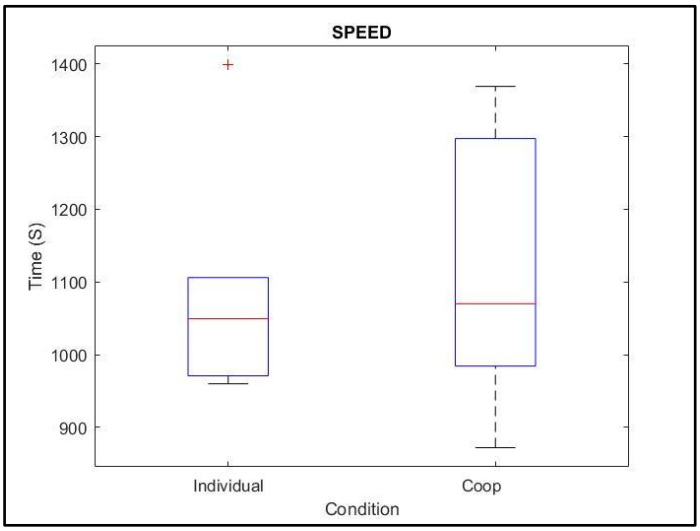


Figure 36 - Boxplot of the time (in seconds) it took to complete the game between the individual and cooperative conditions

The boxplot in figure 36 conveys that students were quicker in an individual DGBL environment (lower value indicates a quicker time in seconds (s)) as the median, lower quartile, and upper quartile is of a lower value. However, the quickest speed was achieved in a cooperative DGBL environment.

Cooperation - Physical Cooperative DGBL vs. Virtual Cooperative DGBL

Comparing cooperation in physical cooperative DGBL and virtual cooperative DGBL resulted in a significance value of $p = 0.43$; thus, resulting in the acceptance of the null hypothesis for factor **a** of hypothesis 2.

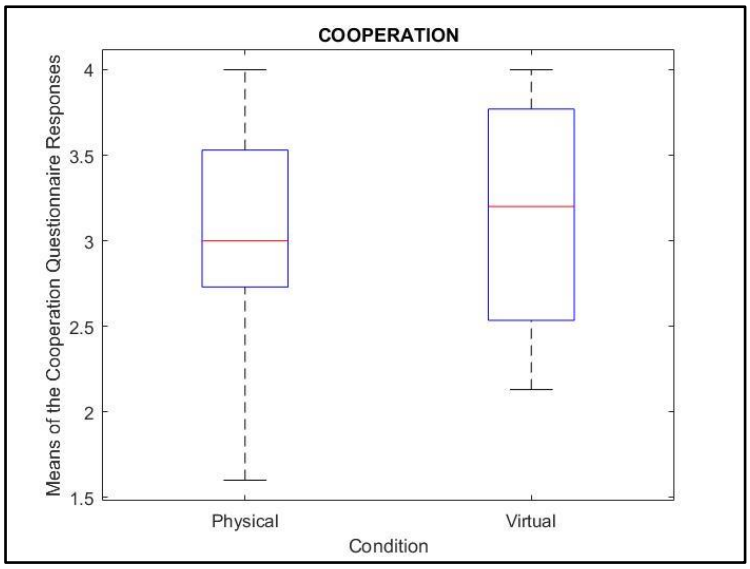


Figure 37 - Boxplot of means of cooperation questionnaire relating to cooperation

Figure 37 suggests that effective cooperation is most likely to occur in a virtual cooperative DGBL environment due to a higher median, upper quartile and minimum value; although both conditions show that they facilitate a highly cooperative environment.

Precision - Physical Cooperative DGBL vs. Virtual Cooperative DGBL

Comparing precision in physical cooperative DGBL and virtual cooperative DGBL resulted in a significance value of $p = 0.75$; thus, resulting in the acceptance of the null hypothesis for factor **b** of hypothesis 2.

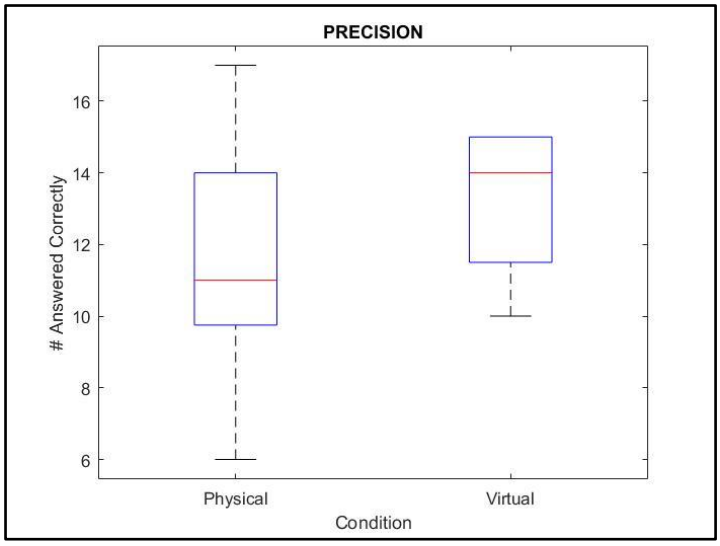


Figure 38 - Boxplot of correctly answered mathematics questions between the physical and virtual mediums

The boxplot in figure 38 shows that participants were more consistent with providing more correct answers when cooperating over a virtual medium; as seen by a higher median, upper quartile, and minimum value. However, cooperating over a physical

medium produced the most precise attempt, with the highest maximum value of 17 (100%).

Speed - Physical Cooperative DGBL vs. Virtual Cooperative DGBL

Comparing speed in physical cooperative DGBL and virtual cooperative DGBL resulted in a significance value of $p = 0.68$; thus, resulting in the acceptance of the null hypothesis for factor **c** of hypothesis 2.

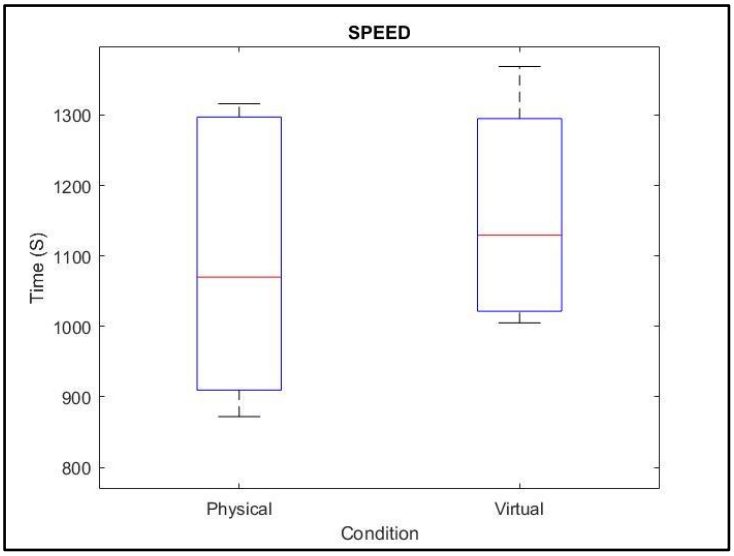


Figure 39 - Boxplot of the time (in seconds) it took to complete the game between the physical and virtual mediums

Furthermore, figure 39 indicates that students were quicker in a physical cooperative DGBL as the minimum, median, lower quartile, and maximum values all indicate this.

Qualitative Results

Motivation

11 motivation-related questions were included in both the individual and cooperative questionnaire to gauge and compare student's motivation in an individual learning environment and a cooperative learning environment, whilst also altering the cooperation medium in the cooperative learning environment. Seven of the 11 questions were answered using a four-point Likert scale. The reasoning for utilising a four-point Likert scale was to ensure participants could not select a neutral response, thus only allowing them to select from "Very easy", "Easy", "Hard", and "Very hard" for example. Whilst the remaining four questions were answered using Boolean values of "Yes" or "No".

The means of all mean motivation questionnaire responses were calculated for all three test conditions (see figure 40):

- **Individual Condition** - Mean = 3.59
- **Cooperative Condition (Physical & Virtual Combined)** - Mean = 3.44

- **Cooperative (Physical) Condition** - Mean = 3.40
- **Cooperative (Virtual) Condition** - Mean = 3.48

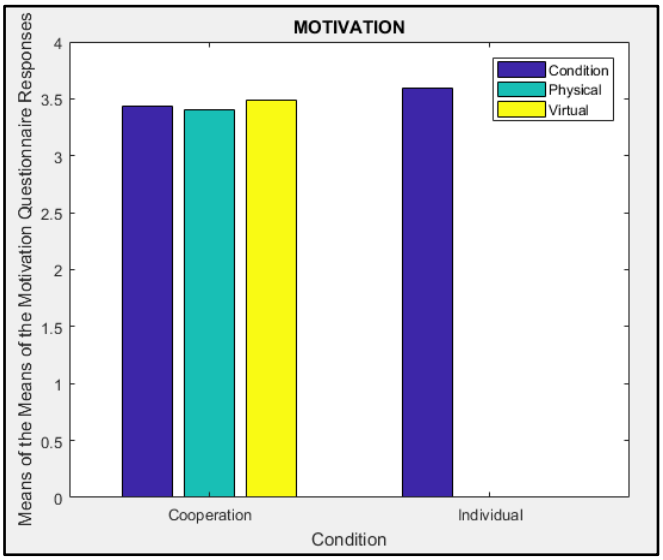


Figure 40 - Comparing the means of condition specific means for motivation-related questionnaire responses.

In the individual learning environment two participants commented *"I thought it was very difficult but I did very well"* and *"It was a fun game and I became very concentrated,"* conveying a high level of motivation. In the cooperative condition, all participants responded with *"Strongly agree"* in regards to the question "Would you like to learn other school subjects by playing similar games" also supporting a high level of motivation in a GBL environment. Furthermore, all participants either agreed or strongly agreed with the statement "The game was fun to play," again showing a high degree of motivation for the learning experience.

Observations were made in all test conditions regarding how motivated participants were to explore game elements and simply just enjoy the game experience, and how motivated they were to understand and learn probability from the learning content within the game. In the individual condition, attentive reading of each probability question, and answer questions in a timely manner were considered as actions which showed an interest in the learning content. Whereas, in the cooperative condition in addition to what was mentioned previously, behaviour such as discussing answers, contributing ideas, and collaboration that was relevant towards the learning content were also considered as actions that showed an interest in learning. On the other hand, actions that conveyed motivation towards the game experience were characterised as the individual being meticulous when reading the narrative, showing satisfaction when experiencing game milestones (e.g. building a house), using more than the average time to explore the virtual environment, indicating a focus on exploration and not the current task at hand. With these observations in mind, it was concluded that in the case of an Individual GBL environment participants were more motivated by the learning content than the game experience. In contrast, in a Cooperative GBL environment participants were more motivated by the game experience, rather than the learning content.

Cooperation

15 cooperation-related questions were introduced in the cooperative questionnaire with the objective to determine whether cooperative actions were more commonly performed in a physical cooperative environment or virtual cooperative environment. All cooperation questions were answered using the four-point Likert scale. The means of all mean cooperation questionnaire responses were calculated for both cooperation conditions (see figure 41):

- **Cooperative (Physical) Condition** - Mean = 3.01
- **Cooperative (Virtual) Condition** - Mean = 3.14

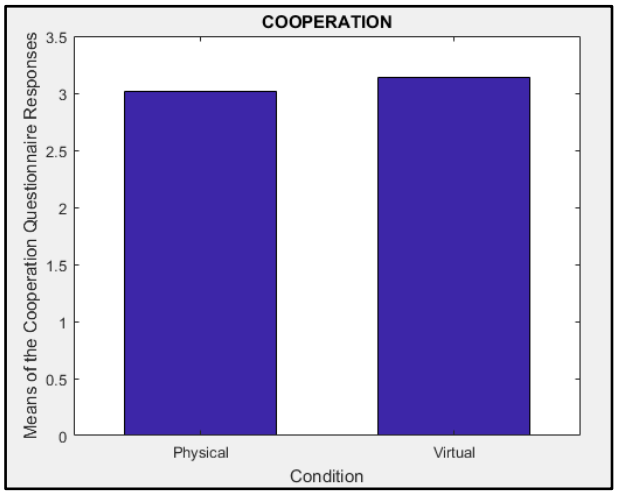


Figure 41 - Comparing the means of physical cooperation and virtual cooperation for cooperation-related questionnaire responses.

Only one group from the physical cooperative condition managed to answer all 17 questions correctly. One participant from the successful group was observed as being passive to begin with, eventually leading to shouting his answers at his partner aimlessly. The same individual expressed *"My partner was good but he did not let me do that much"*, which could possibly be due to frustration and lack of cooperation from his partner. On the other hand, the other participant was observed to be focused and filling a leadership role in the team. A common observation on both individuals was that they skipped most of the narrative text, and were much more attentive towards the probability questions. In contrast the worst performing group participated in the virtual cooperative condition, only managing to answer six out of 17 questions correctly. Although, one participant disclosed that they did not like computer games, whereas the other was observed as showing a lack of interest in the experience, contrary to his positive response to the questionnaire question *"Was it fun to play the game."* Although, a common observation regarding the bad performance between the two participants was that both were silent for majority of the test session, only executing a total of three observed cooperative actions between them; suggesting that a lack of cooperation (and motivation) resulted in a lacklustre result.

For both cooperative conditions, participants responded with *"We did well because we worked together"*, *"We worked together"*, *"We listened to each other,"* in regards to the

question *"Why did you do good?"* thus indicating that in some cases the process of cooperation was perceived to be successful. Most recognisably, the cooperation questionnaire conveyed that over 94% of participants either "Agree" or "Strongly Agree" that they listened to their partner. Additionally, over 88% of participants shared information with their partner if they found out something new, whereas only 53% of participants asked their partner for new information. Moreover, only as little as 50% of participants suggested a way to fix a problem, which aligned with the cooperative actions observed. Concluding, 33% and 20% of participants were strongly reluctant in changing their plans, or asking for help when needed, respectively.

Further observations were made regarding how cooperative and communicative participants were in their respective groups. It was observed that in the physical cooperative condition the following participants were classified: two dominant characters, four passive characters, and four well-balanced characters. In the virtual cooperative condition the following classifications were specified: two dominant characters, two passive characters, four well-balanced characters.

An interesting observation relating to the correlation of cooperation and nature of character was that in 66% of cases where a group possessed a passive individual, another individual would seize the role of leader to initiate cooperation; and 66% of participants whom were perceived as passive were male, and only 25% of individuals whom seized the leadership role were male, thus illustrating that females are more inclined to lead when required to.

Individualism-Collectivism

Test participants whom received the Individual and Cooperative questionnaire gave opinions on statements that were relevant to the taxonomy of Individualism-Collectivism. The questions were adapted from Wagner's (1995) questionnaire of Individualism-Collectivism and assessed the following: personal independence, self-reliance, competitive success, the value of working alone, personal interests, group needs, personal pursuits and group productivity.

The questions were answered on the following scale: 1 = Strongly Disagree and 4 = Strongly Agree. Furthermore, the following questions were reverse-coded (and thus reverse scored) to ensure the goal of the questionnaire remained obscure to the participant: *"A person does better when he works alone," "You can only count on yourself," "It annoys me when other people do better than me,"* as obviously, a strong agreement with the following statements indicated traits of individualism. Individualism-Collectivism was classified by comparing means of responses against the median of the Individualism-Collectivism questions in the questionnaire (median = 2.5). A collectivist was indicated when the total mean of the given response was higher than the median of the questionnaire; contrastingly an individualist was nominated when the total mean of the given response was lower than the median of the questionnaire, and in some cases when the total mean of the given response was the same as the median of the questionnaire, a *"neutralist"* was indicated. Therefore, based on these classification guidelines the following results were acknowledged (see figure 42 & 43):

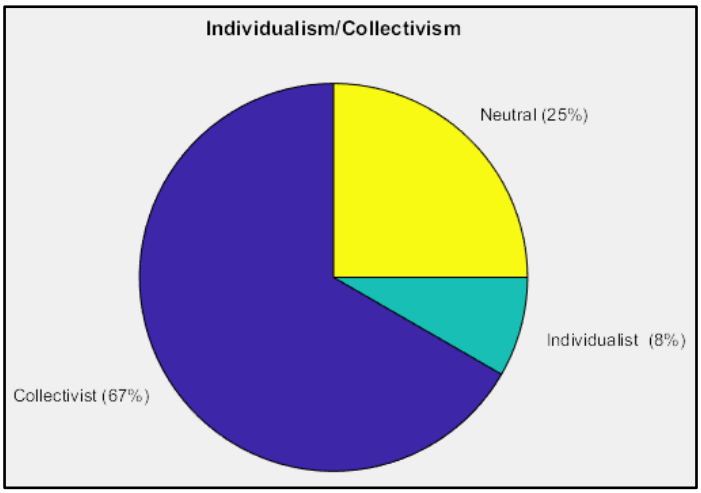


Figure 42 - The results of the taxonomy of Individualism/Collectivism (n = 24).

In addition, figure 43 illustrates Individualism/Collectivism, and conveys the gender breakdown of collectivists, individualists and neutralists.

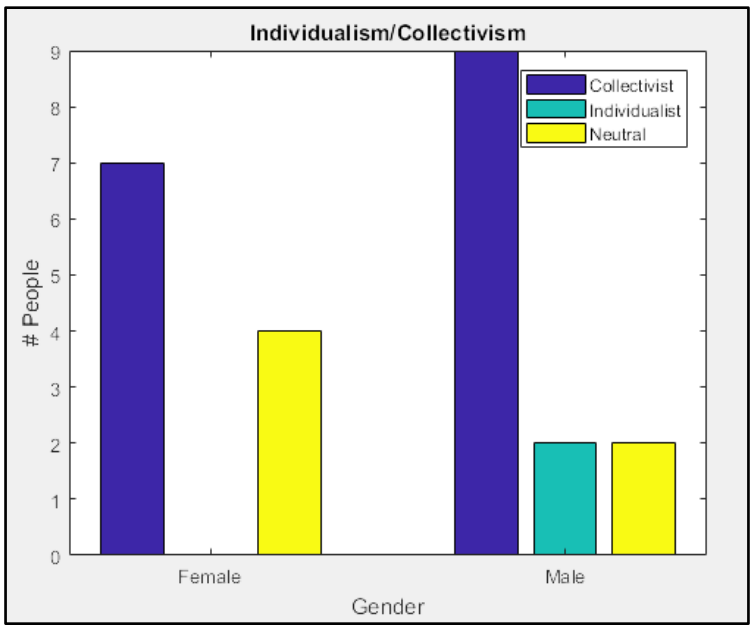


Figure 43 - A representation of gender spread Individualism-Collectivism (n = 24).

Observations were recorded during the experiment in both cooperative conditions (physical and virtual). The degree of cooperation and cooperative effort an individual conveyed was determined dependent on how many of the eight common cooperative actions an individual executed (cooperative actions inspired from Gillies and Boyle (2013). More specifically the mean of the cooperative actions performed were compared against the median of all cooperative actions (median = 0.5), if the mean resulted in a value higher than the median then the cooperative actions executed by the individual was considered to support a collectivist attitude. In contrast, a mean value lower than the median suggested that the lack of actions executed represented an individualist attitude. A mean value equal to the median implied that the actions performed by the attitude were not significant enough to categorise them as a collectivist or individualist,

but rather a neutralist. See figure 44 for number of executions by participants for the following cooperative actions: contribute ideas, explain concepts, providing examples to add to or illustrate an idea, ask a question to promote conversation, give reasons for an answer, provide suggestion for approaching a task, build on ideas of others in groups, and affirming ideas of others.

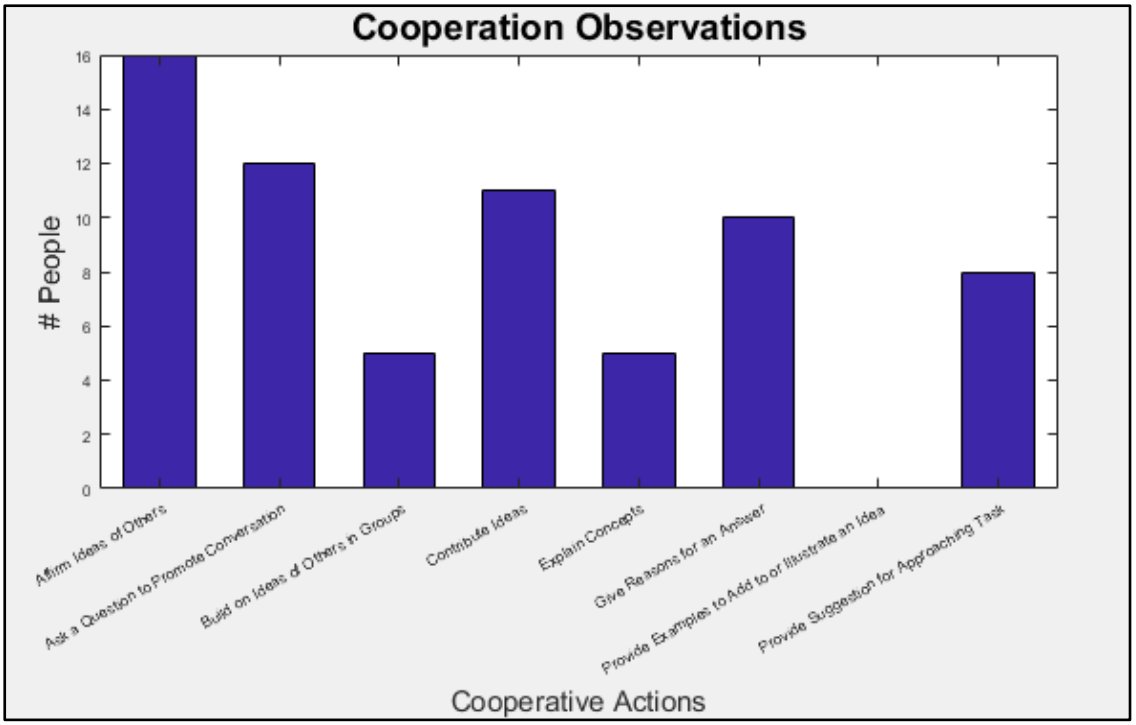


Figure 44 - A representation of cooperative actions performed in a cooperative (physical and virtual) GBL environment (n = 18).

Therefore, based on these results the most common cooperative activities that were performed by at least 50% of participants were:

- The contribution of ideas.
- Asking questions to promote conversation.
- Giving reasons for answers.
- Providing a suggestion to approach a task.
- Affirming the ideas of others.

Whereas the following activities were only performed by as few as only 36% of participants:

- Explain Concepts
- Build on Ideas of Others in Groups

From the observations, it can be assumed that the most primitive activity of cooperation is the affirmation of ideas by others as this was the most common action performed by participants, with 88% of participants performing this action in a cooperative environment. In contrast, the activity of providing examples to add to or help illustrate an idea was performed by no individual, and thus can be determined as a task that

illustrates a high level of awareness in understanding the group's needs and is probably an advanced level of cooperative action. Out of the 12 classified collectivists that participated in the cooperative condition, only five participants executed enough cooperative actions to support the attitude and actions of a collectivist. Out of the remaining seven classified collectivists, four were observed as illustrating an individualist attitude due to their lack of cooperative actions, and the remaining three were recognised as neutralists.

From the single classified individualist, the participant was perceived as a neutralist based on the number of cooperative actions performed. From the previously classified 5 neutralists, based on the number of cooperative actions performed, two were aligned and were classified as a neutralist, although the remaining three were classified as possessing an individualist attitude (see appendix C2).

Further observations indicated that there was no correlation between someone being proactive and taking the role of leader and the classification of Individualism-Collectivism (regarding the questionnaire). Furthermore, there was no correlation between someone who chose to participate in a follower role and the classification of Individualism-Collectivism. Although there was correlation between someone who was observed as being rather silent during the experiment and the mean number of cooperative actions executed; individuals whom were described as very quiet had a mean value of 0.13 and 0.25 respectively.

Discussion

Quantitative data indicated that an individual GBL environment was more effective than a cooperative GBL condition, and that cooperating over a virtual medium was more effective in enhancing the process of cooperation than when cooperating in a common physical space. The significance values for all dependent variables (motivation; $p = 0.67$, precision; $p = 0.18$, speed; $p = 0.43$) of hypothesis 1 demonstrate that it cannot be confirmed whether a cooperative DGBL environment was more effective than an individual DGBL environment. Because of the results of the statistical tests the alternative hypothesis for hypothesis 1 was rejected; accepting the following null hypothesis: *"An individual GBL environment is just as effective as a cooperative GBL environment."* Figure 34 and 36 also support the null hypothesis and convey that students were quicker and more motivated in an individual DGBL environment than in a cooperative DGBL environment. However, figure 35 supports the fact that students achieved more correct answers in a cooperative DGBL environment. The significance values for all dependent variables (cooperation; $p = 0.80$, precision; $p = 0.75$, speed; $p = 0.68$) for hypothesis 2 suggest that cooperation in a common physical space is no more effective than cooperating over a virtual medium, and therefore the alternative hypothesis for hypothesis 2 must also be rejected; accepting the null hypothesis - *"The process of cooperation is just as effective when using a common physical space or a virtual medium."* Figure 37 and 38 illustrate that students were more cooperative and achieved more correct answers when cooperating over a virtual medium than that of a common physical space.

Supplementary qualitative data was gathered via observations and mini-focus group interviews. By performing observations, it was possible to evaluate whether game design and mechanics functioned cohesively. Cohesive game design was gauged by familiarity and understanding of the following factors: game controls, game mechanics, UI information, and the ability to find game collectables in a reasonable time. During observations, we should have been aware of the *"Hawthorne Effect"* (where test participants potentially perform better or worse as a result of being watched) and the influence the phenomenon had on our observations. We should have lessened the likelihood of the Hawthorne effect by consistently performing discrete observations. After each test session, a brief mini-focus group interview was organised (composed of two or three students, with a duration of approximately two minutes) with the intent to obtain opinions from participants in regards to: the difficulty level of the learning content, how enjoyable the experience of GBL was, and if the narrative was coherent and easy to grasp. In each mini-focus group interview we respected the likelihood of bias and group conformity influencing the answers of participants. Thus, when receiving an answer from individual 1 after an answer was given from individual 2, verification type questions were directed towards individual 1 to ensure the opinion was genuine, and it was not just an attempt to conform to another individual's opinion. The general impression from focus group responses was that the math questions were of appropriate difficulty, easy to begin with and became progressively harder, thus supporting a fundamental element of a generic *"flow"* experience. This impression was further validated by evaluating the number of participants that correctly answered specific questions during the test. Results showed that 91% of participants answered the first question correctly, whereas only as few as 29% answered the last question correctly, possibly illustrating an appropriate challenge curve, supporting the perception of challenge alignment in flow (Csikszentmihalyi, 1975). The suitability and correlation between the existing challenge curve and one's skill may vary from each individual due to the fluctuating nature of competency amongst students.

Feedback received regarding gameplay from the mini-focus group interviews yielded satisfactory results; a majority of participants commented with *"simple"* and *"easy to use."* However, a minority of participants were observed as struggling, at least to begin with; but from their responses to the questionnaire it was not so surprising, as those participants were classified as unfamiliar with video games. Game design related observations indicated that on multiple occasions three participants attempted to click the animated quest icon, rather than the NPC when attempting to obtain or finish the current quest. Two participants were also oblivious to the fact that when an item was highlighted in orange, it indicated that it needed to be interacted with. In one cooperative test condition, both participants were having issues with obtaining and delivering quests since there was only one common interaction point attached to an NPC, as both player prefabs possessed a collider component, it resulted in both player's blocking each other from the common interaction point. In a similar occasion two participants enjoyed the fact that they could influence the other player's movement and nudge the other player away from the interaction point. Feedback received regarding narrativity was mostly positive. The majority of participants provided feedback which

indicated a coherent understanding of the intended narrative, thus also reflecting a high level of narrative closure. However, some participants expressed that they were simply not interested in the narrative, and thus they skipped the textual information when it was presented, thus their comprehension of the narrative was dubious to say the least.

Upon evaluating the design of "Plain Probability" after testing, we suggested that an introductory cut scene could be implemented to elaborate on the control scheme and backstory that existed in Plain Probability. By doing so it may lessen the number of frustrated students who were new to video games and DGBL. Secondly, it may better define the purpose and motivation to participate in the game's narrative. Immediate feedback could be enhanced by implementing the instantiation of particles at the point where the terrain was clicked on the NavMesh, to provide the player with a spatial-visual UI element informing the player of the avatar's current target destination. Additionally, if the area of the terrain that was clicked on by the player is non-traversable by the avatar, then the instantiated particles could be colour-coded as green or red implicitly conveying if the clicked-on area is traversable by the avatar. The method in which dialogue was presented in the game resulted in some issues of overwriting the current narrative text when a player clicked an NPC twice (e.g. on a double click). A way to combat this could be to implement a timer threshold between clicks, so clicking twice within a second would not close the current dialogue. Alternatively, a button could be added that would toggle a dialogue box conveying a history of all previous dialogue. To improve further on player autonomy in a cooperative setting, the game could be designed to allow each player to select a unique answer for the current question, although it could be argued that the current structure of shared answers supports positive interdependence and cooperation as both players must come to an agreement before submitting an answer.

There were indications that Plain Probability succeeded in a DGBL environment. The test participants found Plain Probability to be motivating and engaging, enjoying exploration and the narrative. This illustrates a successful use of: Garris et al.'s (2002) IPOGM, Chiasson and Gutwin's (2005) suggestions on designing for child development, Deci and Ryan's (2000) factors of self-determination theory, incorporating Schoenau-Fog's (2011) OA3 framework, and Malone's (1980) understanding of curiosity. Plain Probability sustained the game process cycle by presenting immediate feedback in the form of narrative text and rewards. Accomplishment of exploration and experiencing activities facilitated the maintenance of intrinsic motivation. Exploration related activities satisfied sensory curiosity, and experiencing related activities satisfied cognitive curiosity. An increasing challenge curve granted individuals with the sensation of self-competence, also inducing flow. The need for player autonomy was fulfilled when experiencing exploration activities and answering question. Lastly, physical development was aided by implementing a satisfyingly simple user interface.

Conclusion

The purpose of this thesis was to provide further insights relating to cooperative learning within a digital domain, and an investigation of whether the process of cooperation can be augmented if the cooperation medium that hosts the cooperative activity is altered. The incorporation of a narrative fantasy in Plain Probability was inspired by Garris et al.'s (2002) IPOGM, ensuring that the activity of learning is made more intrinsically motivating by giving learning content a fictional context that learners can relate to, and find more interesting.

24 test subjects participated in the two-phased experiment, where they experienced one of three conditions: *individual DGBL, physical cooperative DGBL, or virtual cooperative DGBL*. Data was gathered in qualitative and quantitative formats from observations, short interviews, a questionnaire and the generated text file from the gameplay session. The data illustrated the success of the participants learning experience, their enjoyment of the learning experience, and (in relevant test conditions) how cooperative they were in a cooperative learning environment. The text file provided objective-related ratio data, such as the time of the game session and the number of correctly answered questions. Questionnaires, interviews and observations provided additional data of subjective nature relating to a participant's interest and motivation towards the experience, as well as their self-perceived cooperative efforts. After scrutiny of all qualitative and quantitative data, it could be concluded that an individual DGBL environment was the most effective environment, and that a virtual cooperative medium provided more effective cooperation, although none of these subsequent statements were statistically proven. Anderson-Darling tests were executed on all data sets to test for normality. These results influenced the decision to use either a rank sum or two-sample t-test when determining statistical significance of each dependent variable for both hypotheses. Unfortunately, the alternative hypotheses for all dependent variables were rejected due to the insignificant results of the rank sum and two-sample t-tests.

Research Question 1

Accept Null Hypothesis

"An individual GBL environment is just as effective as a cooperative GBL environment."

Results: Motivation - P-value = 0.67. Precision - P-value = 0.18 Speed - P-value = 0.43

Research Question 2

Accept Null Hypothesis

"The process of cooperation is just as effective when using a common physical space or a virtual medium."

Results: Cooperation- P-value = 0.80. Precision - P-value = 0.75 Speed - P-value = 0.68

As the sample size of the experiment was rather limited, the rejection of the alternative hypotheses could be viewed as a potential motive for further investigation, due to the fact that the data set is more prone to skewing when outliers are present within it, which also reduces the likelihood of experimental success. Acquiring more test participants would make it possible to distinguish between individuals who participated in the virtual/physical cooperative condition, and those who participated in the cooperative DGBL condition. It would allow us not to assume that participants who participated in the virtual/physical cooperative condition also participated in the cooperative DGBL condition. Reflecting upon the test approach, some motivation and cooperation related questionnaire questions could have been reverse coded to disguise the goal of the experiment better. In some cases, the intention of some questions could have been made more obvious to avoid ambiguity. For example, the "Winning is everything" statement in the individualism-collectivism questionnaire does not necessarily define a collectivist or individualist, and therefore the score for that question was excluded from the individualism-collectivism classification.

Plain Probability succeeded in achieving a motivating learning environment as the application successfully incorporated most theories, frameworks and models presented previously, despite the test results failing to endorse the alternative hypotheses.

In the future, it could be interesting to investigate what impact positive goal interdependence and negative goal interdependence have on learning experiences, and whether a cooperative or competitive DGBL environment constitutes towards a more successful learning environment. Additionally, it could be interesting to investigate whether a competitive attitude is intensified when competitors are competing in a common physical space or virtual medium.

References

Admiraal, W., Huizenga, J., Akkerman, S., & Dam, G. ten. (2011). The concept of flow in collaborative game-based learning.

Andrews, M. (2010). Game UI Discoveries: What Players Want.

Bandura, A., & Schunk, D. H. (1981). Cultivating competence, self-efficacy, and intrinsic interest through proximal self-motivation.

Barzilai, S., & Blau, I. (2014). Scaffolding game-based learning: Impact on learning achievements, perceived learning, and game experiences.

Berlyne, D. (1960). *Conflict, Arousal, and Curiosity*.

Berlyne, D. (1965). *The Structure and Direction in Thinking*.

Bilash, O. (2009). Inductive and Deductive Instruction.

Bjørner, T., & Forlag, H. R. (2015). *Qualitative Methods For Consumer Research*.

Brown, E., & Cairns, P. (2004). A Grounded Investigation of Game Immersion.

Bruni, L. E., & Baceviciute, S. (2013). Narrative Intelligibility and Closure in Interactive Systems.

Caillois, R. (1961). *Man, Play and Games*.

Chiasson, S., & Gutwin, C. (2005). Design Principles for Children's Technology.

Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice.

Crookall, D., Oxford, R., & Saunders, D. (1987). Simulation/Games For Learning: Towards A Reconceptualization Of Simulation: From Representation To Reality.

Csikszentmihalyi, M. (1975). *Beyond Boredom and Anxiety*.

Csikszentmihalyi, M. (1996). CREATIVITY: Flow and the psychology of discovery and invention.

Dasgupta, A. (1998). Which Mammal lays Eggs?

Deci, E., & Ryan, R. (2000). The' what' and' why' of goal pursuits: Human needs and the self-determination of behavior.

Deutsch, M. (2001). Cooperation and Competition.

Druckman, D. (1995). The educational effectiveness of interactive games, In D. Crookall & K. Arai (Eds.).

Ermi, L., & Mäyrä, F. (2005). Fundamental Components of the Gameplay Experience: Analysing Immersion.

Fisher, C. (1978). The effects of personal control, competence and extrinsic reward systems on intrinsic motivation.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model.

Gillies, R., & Boyle, M. (2013). Cooperative Learning: A Smart Pedagogy for Successful Learning.

Gilmore, B., Pine, J., & James, H. (1999). *The Experience Economy*.

Gredler, M. (1992). Designing and Evaluating Games and Simulations: A Process Approach.

Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning.

Jensen, H. N., Lindhardt, B., Møller, M. T., Andersen, M. W., & Weng, P. (2006). *KonteXt 3 Elevbog*.

Johnson, D. W., Johnson, R. T., & Smith, K. (2013). Cooperative Learning: Improving University Instruction By Basing Practice On Validated Theory.

Kapp, K. (2014). GAMIFICATION: Separating Fact From Fiction.

Kashdan, T. B., Rose, P., & Fincham, F. D. (2004). Curiosity and Exploration: Facilitating Positive Subjective Experiences and Personal Growth Opportunities.

Kiili, K. (2006). Digital game-based learning: Towards an experiential gaming model.

Lahri, P. (2015). Games vs Game Based Learning vs Gamification.

Lawyerment. (2017). What is Motivation? What are the Concepts of Motivation?

Madsen, J., Anesen, L., & Winther, N. (2008). *Format 3 Elevbog*.

Malone, T. W. (1980). What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games.

McIntyre, J. M., & Rubin, I. M. (1971). *Organizational psychology: An experiential approach.*

Meluso, A., Meixun, Z., Hiller, S. A., & James, L. (2012). Enhancing 5th graders' science content knowledge and self-efficacy through game-based learning.

Nussbaum, M. (2010). Games, Learning, Collaboration and Cognitive Divide.

Pierfy, D. (1977). Comparative simulation game research: Stumbling blocks and stepping stones Simulation and Games.

Ricci, K., Salas, E., & Cannon-Bowers, J. (1996). Do computer-based games facilitate knowledge acquisition and retention?

Rieber, L. P. (1991). Animation, incidental learning, and continuing motivation.

Rieber, L. P. (2001). Designing learning environments that excite serious play.

Ryan, M.-L. (2001). *Narrative as Virtual Reality: Immersion and Interactivity in Literature and Electronic Media*.

Ryan, M.-L. (2006). *Avatars of Story*.

Ryan, R., Stiller, J., & Lynch, J. (1994). Representations of relationships to teachers, parents, and friends as predictors of academic motivation and self-esteem.

Sawyer, B. (2002). Serious Games: Improving public policy through game-based learning and simulation.

Schoenau-Fog, H. (2011). The player engagement process–An exploration of continuation desire in digital games.

Tsai-Sun, & Wang. (2011). Game Reward Systems: Gaming Experiences and Social Meanings.

Unity. (2016). *Adventure Game Tutorial*. Retrieved from https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial

Vallerand, R. J., & Fortier, M. S. (1997). Self-Determination and Persistence in a Real-Life Setting: Toward a Motivational Model of High School Dropout.

Wagner, J. A. (1995). Studies of individualism-collectivism: effects on cooperation in groups.

Wolters, C. . (1998). Self-regulated learning and college students' regulation of motivation.

Zuckerman, M., Porac, J., Lathin, D., & Deci, E. L. (1978). On the Importance of Self-Determination for Intriniscally-Motivated Behavior.

Appendix A

1. Comparing games, gamification, and GBL (Lahri, 2015).

Factor	Games	Gamification	Game-Based Learning
Purpose	Purely for pleasure. Rules and objectives may not be defined.	May be a group of tasks with assigned points or rewards.	Learning objectives are defined.
Competition	Winning and losing are distinct states.	In some cases losing may not be possible as the objective is to motivate people to participate and take action.	In some cases losing may not be possible as the objective is to motivate people to participate and learn.
Motivation	Gameplay is the primary motive, rewards are a secondary motive that supplement the gameplay.	Can be intrinsically rewarding.	Occasionally playing the game can be intrinsically rewarding.
Cost	Difficult and very expensive to make.	Usually cheaper to implement.	Difficult and very expensive to make.
Content	Games naturally incorporate game characteristics and narratives, as part of the game experience.	Usually game characteristics are built on top of existing learning content.	New learning content is merged with game characteristics, to form a cohesive game-based experience.

2. The final score screen in Plain Probability, introducing reflective learning and debriefing.

Appendix B

1. The method for checking condition satisfaction.

```
public static bool CheckCondition(Condition requiredCondition)
{
    Condition[] allConditions = Instance.conditions;
    Condition globalCondition = null;

    if (allConditions != null && allConditions[0] != null)
    {
        for (int i = 0; i < allConditions.Length; i++)
        {
            if (allConditions[i].hash == requiredCondition.hash)
                globalCondition = allConditions[i];
        }
    }

    if (!globalCondition)
        return false;

    return globalCondition.satisfied == requiredCondition.satisfied;
}
```

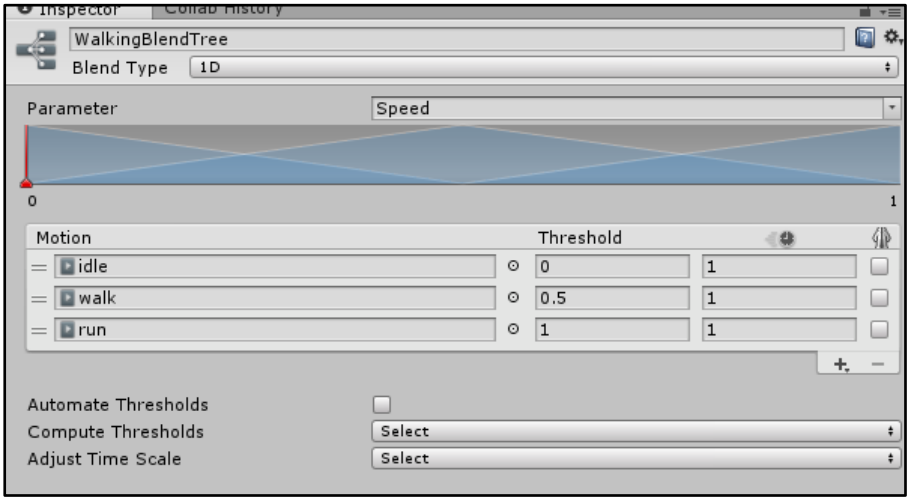
2. The ConditionCollection class.

```
public class ConditionCollection : ScriptableObject {

    public string description;
    public Condition[] requiredConditions = new Condition[0];
    public ReactionCollection reactionCollection;

    public bool CheckAndReact()
    {
        for(int i = 0; i < requiredConditions.Length; i++)
        {
            if (!AllConditions.CheckCondition(requiredConditions[i]))
                return false;
        }


        if (reactionCollection)
        {
            reactionCollection.React();
        }


        return true;
    }
}
```

3. The ReactionCollection class.

```
public class ReactionCollection : MonoBehaviour
{
    public Reaction[] reactions = new Reaction[0];

    private void Start ()
    {
        for (int i = 0; i < reactions.Length; i++)
        {
            DelayedReaction delayedReaction = reactions[i] as DelayedReaction;

            if (delayedReaction)
                delayedReaction.Init ();
            else
                reactions[i].Init ();
        }
    }

    public void React ()
    {
        for (int i = 0; i < reactions.Length; i++)
        {
            DelayedReaction delayedReaction = reactions[i] as DelayedReaction;

            if(delayedReaction)
                delayedReaction.React (this);
            else
                reactions[i].React (this);
        }
    }
}
```

4. The *WalkingBlendTree* blend tree.



5. The Stopping method.

```
private void Stopping(out float speed)
{
    agent.Stop();
    transform.position = destinationPosition;
    speed = 0f;
    if (currentInteractable)
    {
        transform.rotation = currentInteractable.interactionLocation.rotation;
        currentInteractable.Interact();
        currentInteractable = null;
    }
}
```

6. The Slowing method.

```
private void Slowing(out float speed, float distanceToDestination)
{
    agent.Stop();
    float proportionalDistance = 1f - distanceToDestination / agent.stoppingDistance;
    Quaternion targetRotation = currentInteractable ?
        currentInteractable.interactionLocation.rotation : transform.rotation;
    transform.rotation = Quaternion.Lerp(transform.rotation,
        targetRotation, proportionalDistance);
    transform.position = Vector3.MoveTowards(transform.position,
        destinationPosition, slowingSpeed * Time.deltaTime);
    speed = Mathf.Lerp(slowingSpeed, 0f, proportionalDistance);
}
```

7. The Moving method.

```
private void Moving()
{
    Quaternion targetRotation = Quaternion.LookRotation(agent.desiredVelocity);
    transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, turnSmoothing * Time.deltaTime);
}
```

8. The Interactable class.

```
public class Interactable : MonoBehaviour
{
    public Transform interactionLocation;
    public ConditionCollection[] conditionCollections = new ConditionCollection[0];
    public ReactionCollection defaultReactionCollection;

    public void Interact()
    {
        for (int i = 0; i < conditionCollections.Length; i++)
        {
            if (conditionCollections[i].CheckAndReact())
                return;
        }

        if(defaultReactionCollection != null)
            defaultReactionCollection.React();
    }
}
```

9. The Inventory script's fields and AddItem() method

```
public const int numItemSlots = 4;

public Image[] itemImages = new Image[numItemSlots];
public Item[] items = new Item[numItemSlots];

public void AddItem(Item itemToAdd)
{
    for (int i = 0; i < items.Length; i++)
    {
        if (items[i] == null)
        {
            items[i] = itemToAdd;
            itemImages[i].sprite = itemToAdd.sprite;
            itemImages[i].enabled = true;
            return;
        }
    }
}
```

10. The SetupUIInformation() method

```
public void SetupUIInformation()
{
    questName.text = questManager.quests[0].questName.ToString();
    questDesc.text = questManager.quests[0].questDescription.ToString();
}
```

11. The UpdateUIInformation() method

```
public void UpdateUIInformation()
{
    {
        questName.text = questManager.quests[0].questName.ToString();
        questDesc.text = questManager.quests[0].questDescription.ToString();
    }
}
```

12. The ClearQuestUIInformation() method

```
public void ClearQuestUIInformation()
{
    if(questManager.quests.Count <= 0)
    {
        questName.text = "Afslut Spillet";
        questDesc.text = "Gå til tavlen for at afslutte spillet";
        return;
    }

    questName.text = "Find den næste mission";
    questDesc.text = "Gå til udråbstegnet";
}
```

13. The UpdateQuestMarker() method.

```
public void UpdateQuestMarker(QuestDest destination, QuestOrigin origin, SpriteRenderer renderer, SpriteRenderer otherRenderer,
    SpriteRenderer otherRenderer2)
{
    if(questManager.quests.Count <= 0)
    {
        renderer.sprite = null;
        otherRenderer.sprite = null;
        otherRenderer2.sprite = null;
        return;
    }

    if (questManager.quests[0].questState == QuestState.NotAvailable)
    {
        renderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questState == QuestState.Available && questManager.quests[0].questOrigin == origin)
    {
        renderer.sprite = filledExclamationMark;
        return;
    }

    if (questManager.quests[0].questState == QuestState.InProgress && questManager.quests[0].questOrigin == origin)
    {
        renderer.sprite = emptyExclamationMark;
        return;
    }

    if (questManager.quests[0].questState == QuestState.Completed && questManager.quests[0].questDestination == destination)
    {
        renderer.sprite = filledQuestionMark;
        otherRenderer.sprite = null;
        otherRenderer2.sprite = null;
        return;
    }
}
```

14. The UpdateBlackboardQuestMarker() method

```
public void UpdateBlackboardQuestMarker(QuestOrigin origin, SpriteRenderer bRenderer)
{
    if (questManager.quests.Count <= 0)
    {
        bRenderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questState == QuestState.NotAvailable)
    {
        bRenderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questState == QuestState.Available && questManager.quests[0].questOrigin == origin)
    {
        bRenderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questState == QuestState.InProgress && questManager.quests[0].questOrigin == origin)
    {
        if(questManager.quests[0].GetType().Equals(typeof(CollectQuest)) || questManager.quests[0].questDestination.Equals(QuestDest.HousePlot))
        {
            bRenderer.sprite = null;
            return;
        }

        bRenderer.sprite = filledQuestionMark;
        return;
    }

    if (questManager.quests[0].questState == QuestState.Completed && questManager.quests[0].questOrigin == origin)
    {
        bRenderer.sprite = null;
        return;
    }
}
```

15. The UpdateHouseQuestMarker() method.

```
public void UpdateHouseQuestMarker(QuestOrigin origin, SpriteRenderer hRenderer)
{
    if (questManager.quests.Count <= 0)
    {
        hRenderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questDestination != QuestDest.HousePlot)
    {
        hRenderer.sprite = null;
        return;
    }

    if (questManager.quests[0].questDestination == QuestDest.HousePlot)
    {
        if (questManager.quests[0].questState == QuestState.NotAvailable)
        {
            hRenderer.sprite = null;
            return;
        }

        if (questManager.quests[0].questState == QuestState.Available && questManager.quests[0].questOrigin == origin)
        {
            hRenderer.sprite = null;
            return;
        }

        if (questManager.quests[0].questState == QuestState.InProgress && questManager.quests[0].questOrigin == origin)
        {
            if (questManager.quests[0].GetType().Equals(typeof(CollectQuest)) || questManager.quests[0].questDestination.Equals(QuestDest.TownMayor))
            {
                hRenderer.sprite = null;
                return;
            }

            hRenderer.sprite = filledQuestionMark;
            return;
        }

        if (questManager.quests[0].questState == QuestState.Completed && questManager.quests[0].questOrigin == origin)
        {
            hRenderer.sprite = null;
            return;
        }


        if (questManager.quests[0].questState == QuestState.Completed || questManager.quests[0].questState == QuestState.Finished)
        {
            hRenderer.sprite = null;
            return;
        }
    }
}
```

16. The UpdateQuestMarker() method being called in Update().

```
private void Update()
{
    UpdateQuestMarker(QuestDest.TownMayorAssistant, QuestOrigin.TownMayorAssistant,
        TMAquestIconSpriteRenderer, TMquestIconSpriteRenderer, FquestIconSpriteRenderer);
    UpdateQuestMarker(QuestDest.TownMayor, QuestOrigin.TownMayor ,TMquestIconSpriteRenderer,
        TMAquestIconSpriteRenderer, FquestIconSpriteRenderer);
    UpdateQuestMarker(QuestDest.Farmer, QuestOrigin.Farmer, FquestIconSpriteRenderer,
        TMAquestIconSpriteRenderer, TMquestIconSpriteRenderer);
```

17. The ImmediateReaction() method.

```
protected override void ImmediateReaction()
{
    Debug.Log(textManager.GetComponent<PhotonView>().photonView);
    textManager.GetComponent<PhotonView>().photonView.RPC("DisplayMessage", PhotonTargets.All, message, speaker,
        textColor.r, textColor.g, textColor.b, delay);
}
```

18. The DisplayMessage() RPC method.

```
[PunRPC]
public void DisplayMessage (string message, string speaker, float r, float g, float b, float delay)
{
    Color textColor = new Color(r, g, b);
    float startTime = Time.time + delay;
    float displayDuration = message.Length * displayTimePerCharacter + additionalDisplayTime;
    float newClearTime = startTime + displayDuration;

    if (newClearTime > clearTime)
        clearTime = newClearTime;

    Instruction newInstruction = new Instruction
    {
        speaker = speaker,
        message = message,
        textColor = textColor,
        startTime = startTime
    };

    instructions.Add (newInstruction);
    SortInstructions ();
}
```

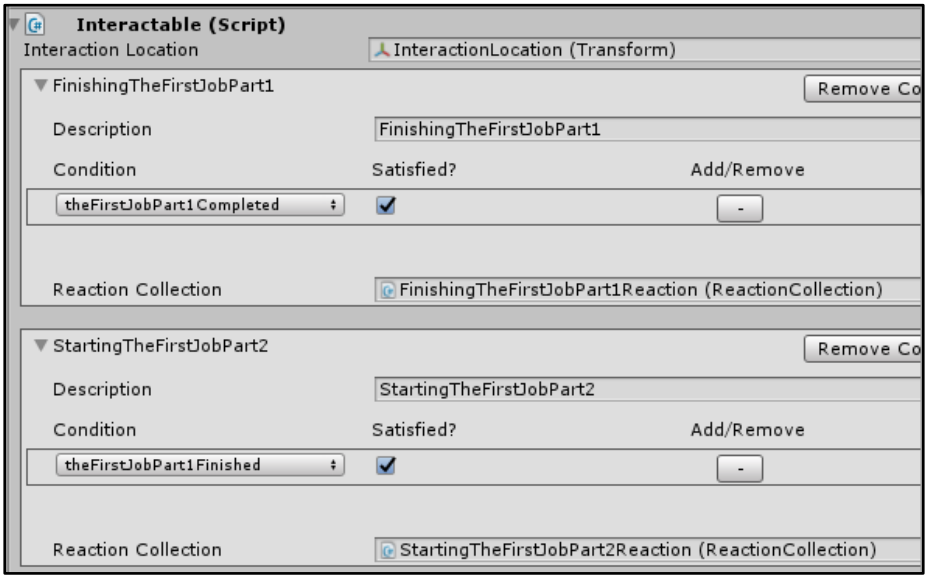
19. The SelectAnswer() RPC method.

```
[PunRPC]
public void SelectAnswerRPC(int i)
{
    uiManager.selectedAnswer = i;
    gameObject.GetComponent<Outline>().enabled = true;
    button1.enabled = false;
    button2.enabled = false;
    button3.enabled = false;
}
```

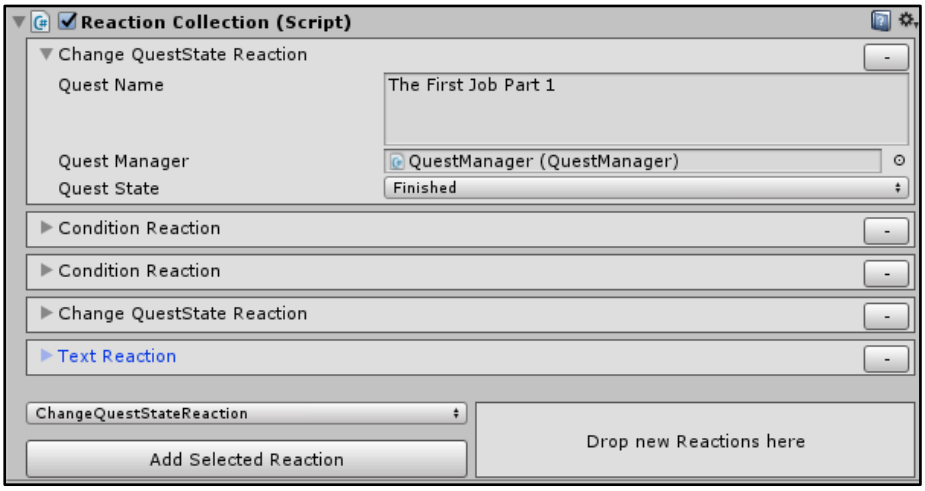
20. The ShowUI() RPC method.

```
[PunRPC]
public void ShowUI()
{
    Debug.Log("interacting with the blackboard...");
    ui.SetActive(true);
    blackboard.SetActive(true);
}
```

21. The UpdateQuestMarker() method being called in Update().



22. A collection of reactions in a reaction collection.



23. An example of an enum property, *QuestState* used in the Quest abstract class.

```
[Description("Quest State")]
public enum QuestState
{
    [Description("Available")]
    Available,
    [Description("Not Available")]
    NotAvailable,
    [Description("In Progress")]
    InProgress,
    Finished,
    Completed,
    NA
}
```

24. The OnJoinedRoom() in the NetworkManager class.

```
public override void OnJoinedRoom()
{
    int playerCount = PhotonNetwork.room.playerCount;
    PlayerCount = playerCount;
    GameObject go = null;
    if (playerCount == 1)
    {
        Vector3 spawn = new Vector3(120.992f, 0, 128.46f);
        go = PhotonNetwork.Instantiate("Player 1", spawn, Quaternion.identity, 0);
    }
    if (playerCount == 2)
    {
        Vector3 spawn = new Vector3(121.992f, 0, 128.46f);
        go = PhotonNetwork.Instantiate("Player 2", spawn, Quaternion.identity, 0);
    }
    if (go != null)
    {
        cameraRig.GetComponent<CameraControl>().playerPosition = go.transform;
    }
}
```

25. Example of calling player movement on an owned PhotonView.

```
if (photonView.isMine)
{

    if (agent.pathPending)
        return;

    float speed = agent.desiredVelocity.magnitude;

    if (agent.remainingDistance <= agent.stoppingDistance * stopDistanceProportion)
        Stopping(out speed);
```

26. Instantiating and stopping a timer in the Update() method.

```
void Update()
{
    if (questManager.finishedQuests.Count >= 1 && startTimer)
    {
    startTimer = false;
    timer = new Stopwatch();
    timer.Start();
    }

    if (!wait)
    {
        wait = true;

        if (questManager.quests.Count <= 0)
        {
            timer.Stop();
        }

        else
        {
            wait = false;
        }
    }
}
```

27. Resetting all conditions in the *AllConditions* asset on game start in the Start() method.

```
void Start ()
{
    AllConditions.Instance.Reset();
    for (int i = 0; i < AllConditions.Instance.conditions.Length; i++)
    {
        if (AllConditions.Instance.conditions[i].hash == Animator.StringToHash("gameStart"))
        {
            AllConditions.Instance.conditions[i].satisfied = true;
            return;
        }
    }
}
```

28. The CalculateCurrentTotalGoldStars() method.

```
public void CalculateCurrentTotalGoldStars()
{
    if (questManager.finishedQuests[questManager.finishedQuests.Count - 1].GetType().Equals(typeof(QuestionQuest)))
    {
        QuestionQuest q = (QuestionQuest)questManager.finishedQuests[questManager.finishedQuests.Count - 1];

        if (q.CheckAnswer(questManager.uiManager.selectedAnswer))
        {
            questManager.uiManager.totalStars++;
            questManager.uiManager.scoreManager.totalGoldStars.text = "x " + questManager.uiManager.totalStars;
            q.questScore = 1;
        }

        questManager.uiManager.selectedAnswer = -1;
    }
}
```

29. The EndGameCalculateTotalGoldStars() method.

```
public void EndGameCalculateTotalGoldStars()
{
    for (int i = 0; i < questManager.finishedQuests.Count; i++)
    {
        subTotalStars = questManager.finishedQuests[i].questScore;
        endTotalStars = endTotalStars + subTotalStars;
        totalStarsResult.text = endTotalStars.ToString();
    }
}
```

30. The CalculateMegaStars() method.

```
public void CalculateMegaStars()
{
    if (questManager.finishedQuests.Count >= 26 && !executeOnce)
    {
        executeOnce = true;

        EndGameCalculateTotalGoldStars();

        theInitiationTotalStars = (questManager.finishedQuests[1].questScore +
            questManager.finishedQuests[2].questScore +
            questManager.finishedQuests[3].questScore + questManager.finishedQuests[4].questScore);
        preparationIsKeyTotalStars = (questManager.finishedQuests[7].questScore +
            questManager.finishedQuests[8].questScore + questManager.finishedQuests[9].questScore +
            questManager.finishedQuests[10].questScore + questManager.finishedQuests[11].questScore +
            questManager.finishedQuests[12].questScore + questManager.finishedQuests[13].questScore +
            questManager.finishedQuests[14].questScore + questManager.finishedQuests[15].questScore +
            questManager.finishedQuests[16].questScore);
        visitingTheFarmerTotalStars = (questManager.finishedQuests[22].questScore +
            questManager.finishedQuests[23].questScore + questManager.finishedQuests[24].questScore);

        DisplayGoldStarResults();

        CalculateTheInitiationMegaStars();
        CalculatePreparationIsKeyMegaStars();
        CalculateVisitingTheFarmerMegaStars();

        totalMegaStarsResult.text = totalMegaStars.ToString();
    }
}
```

31. The DisplayGoldStarResults() method.

```
public void DisplayGoldStarResults()
{
    theInitiationStars.text = theInitiationTotalStars.ToString() + "/4";
    preparationIsKeyStars.text = preparationIsKeyTotalStars.ToString() + "/10";
    visitingTheFarmerStars.text = visitingTheFarmerTotalStars.ToString() + "/3";
}
```

32. The CalculateTheInitiationMegaStars() method.

```
public void CalculateTheInitiationMegaStars()
{
    if (theInitiationTotalStars >= 4)
    {
        totaltheInitiationMegaStars = 3;
        totalMegaStars = totalMegaStars + totaltheInitiationMegaStars;
        theInitiationMegaStar3.SetActive(true);
        return;
    }

    if (theInitiationTotalStars >= 3)
    {
        totaltheInitiationMegaStars = 2;
        totalMegaStars = totalMegaStars + totaltheInitiationMegaStars;
        theInitiationMegaStar2.SetActive(true);
        return;
    }

    if (theInitiationTotalStars >= 2)
    {
        totaltheInitiationMegaStars = 1;
        totalMegaStars = totalMegaStars + totaltheInitiationMegaStars;
        theInitiationMegaStar1.SetActive(true);
        return;
    }

    if (theInitiationTotalStars < 2)
    {
        totaltheInitiationMegaStars = 0;
    }
}
```

33. The CalculatePreparationIsKeyMegaStars() method.

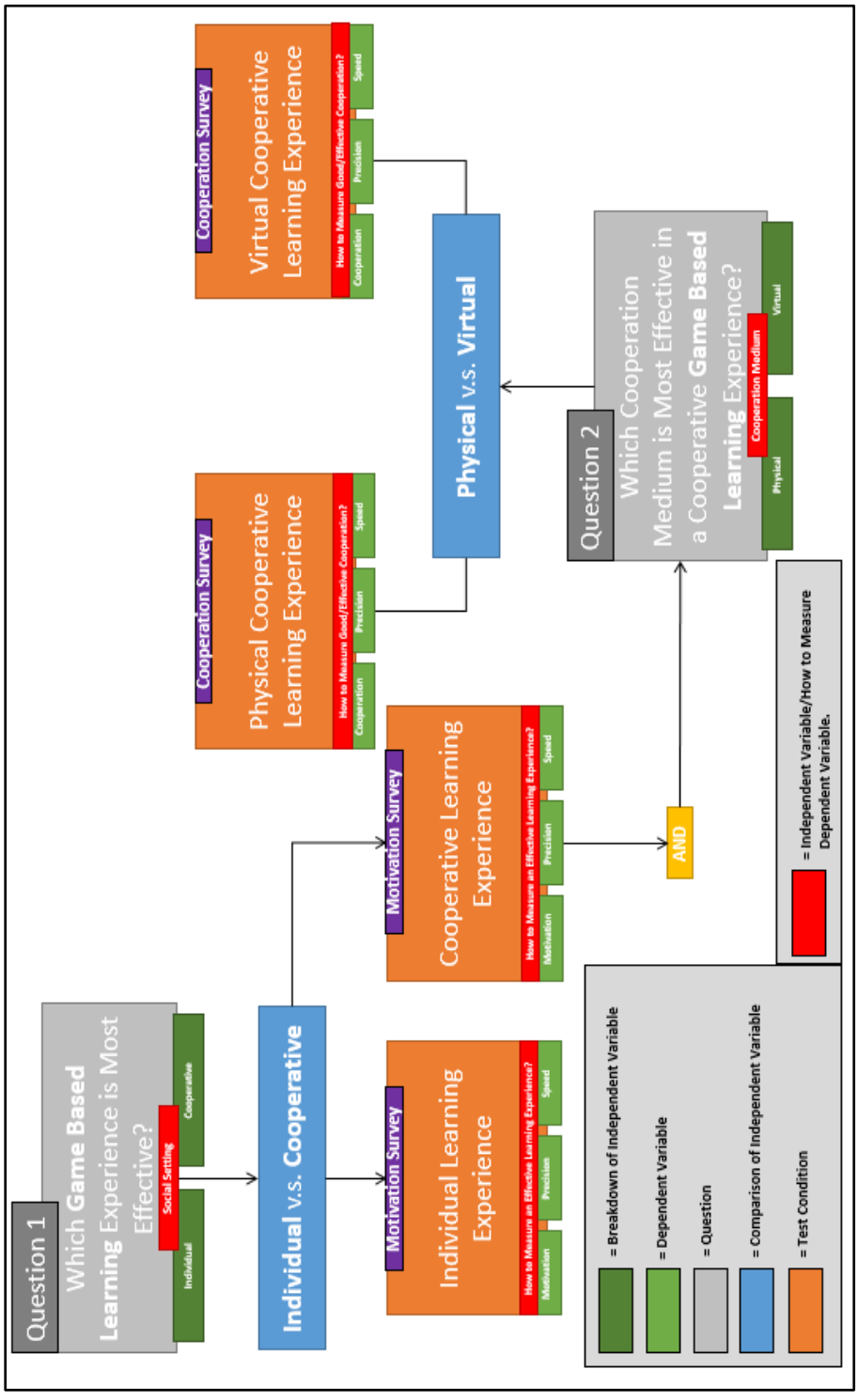
```
public void CalculatePreparationIsKeyMegaStars()
{
    if (preparationIsKeyTotalStars >= 10)
    {
        totalpreparationIsKeyMegaStars = 3;
        totalMegaStars = totalMegaStars + totalpreparationIsKeyMegaStars;
        preparationIsKeyMegaStar3.SetActive(true);
        return;
    }

    if (preparationIsKeyTotalStars >= 8)
    {
        totalpreparationIsKeyMegaStars = 2;
        totalMegaStars = totalMegaStars + totalpreparationIsKeyMegaStars;
        preparationIsKeyMegaStar2.SetActive(true);
        return;
    }

    if (preparationIsKeyTotalStars >= 5)
    {
        totalpreparationIsKeyMegaStars = 1;
        totalMegaStars = totalMegaStars + totalpreparationIsKeyMegaStars;
        preparationIsKeyMegaStar1.SetActive(true);
        return;
    }

    if (preparationIsKeyTotalStars < 5)
    {
        totalpreparationIsKeyMegaStars = 0;
    }
}
```

34. The CalculateVisitingTheFarmerMegaStars() method.

```
public void CalculateVisitingTheFarmerMegaStars()
{
    if (visitingTheFarmerTotalStars >= 3)
    {
        totalvisitingTheFarmerMegaStars = 3;
        totalMegaStars = totalMegaStars + totalvisitingTheFarmerMegaStars;
        visitingTheFarmerMegaStar3.SetActive(true);
        return;
    }

    if (visitingTheFarmerTotalStars >= 2)
    {
        totalvisitingTheFarmerMegaStars = 2;
        totalMegaStars = totalMegaStars + totalvisitingTheFarmerMegaStars;
        visitingTheFarmerMegaStar2.SetActive(true);
        return;
    }

    if (visitingTheFarmerTotalStars >= 1)
    {
        totalvisitingTheFarmerMegaStars = 1;
        totalMegaStars = totalMegaStars + totalvisitingTheFarmerMegaStars;
        visitingTheFarmerMegaStar1.SetActive(true);
        return;
    }

    if (visitingTheFarmerTotalStars < 1)
    {
        totalvisitingTheFarmerMegaStars = 0;
    }
}
```

Appendix C

1. Structuring of test strategy: identification of research questions and independent/dependent variables.



2. The table shows how the previously classified collectivists, individualists, and neutralists (from the questionnaire) align with the classification based on the observations of the number of cooperative actions performed.

Classification	**# Participants**
Previously Collectivist	12
Collectivist	5
Individualist	4
Neutralist	3
Previously Individualist	1
Collectivist	0
Individualist	0
Neutralist	1
Previously Neutralist	5
Collectivist	0
Individualist	3
Neutralist	2