Aalborg University

Medialogy

Simulating Object Stacking Using Stack Stability

Author:
Kasper Kronborg Thomsen

Supervisor:
Martin Kraus

Group:

MTA 141032
Department of Architecture, Design and Media Technology

May 2014

AALBORG UNIVERSITY
DENMARK

AALBORG UNIVERSITY
DENMARK

Department of Architecture, Design and Media Technology
9th and 10th semester

Title: Simulating Object Stacking Using Stack Stability
Semester: 9th and 10th
Semester Theme: Master Thesis
Supervisor: Martin Kraus
Project group: 141032

Kasper Kronborg Thomsen

Abstract:

This project proposes and implements a system for simulating object stacking using stack behavior. The system is implemented as an extension to the physics engine in the game engine Unity. The system's simulation is compared to real objects and the Unity physics engine in several stacking scenarios calculated in real-time. The system shows improvements in most of the scenarios over Unity's physics engine. A user test is also conducted where users compare the proposed system to Unity's physics engine in interactive stacking tasks. The users found the commercial physics engine more visually plausible and found the proposed system more usable.

Finished on: May 25th 2014

Copyright@2014 This report and/or app ended material may not be partly or completely published or copied without prior written approval from the authors. Neither may the contents be used for commercial purposes without this written approval.

Acknowledgements

Thanks to my supervisor Martin Kraus for help and guidance.

Reading Guide

In this report all documentation related to the final system is presented. A supplementary portfolio contains documentation not directly related to the proposed system.

Citations in this report are written as [Author, Year] or as [Company name, year].

A DVD is attached as Appendix A, and contains a digital copy of this report, source code, binaries, digital references, miscellaneous material related to the tests and an AV-Production.

CONTENTS

Chapter 1 Introduction

This project focuses on stacking of simulated wooden blocks in a game physics engine. In this project, stacking and stacks refers to objects placed in deliberate structures. Physics engines for games do not emphasize stacking on small object scales with realistic settings. Small object scales refer to objects with the smallest side at least 1 cm and the largest less than 20 cm. It is therefore suspected that an inability to simulate at these scales is the reason for the lack of games and applications in small object stacking. In the literature, the most common approach to solve problems like this is to develop more advanced models for collision detection, impulse handling and general solvers. This often leads to impressive systems, using advanced math and algorithms to simulate previously difficult scenarios sometimes even in real time. However, these systems require high level knowledge of math and physics to re-implement.

This project presents a different approach to solve the physics problem. Instead of changing the underlying physics model, it describes a system that corrects the normal physics system to enable simulation of stacking behavior. The system was implemented in a commercial game engine and tested on adult users with novice to intermediate knowledge of physics systems. A comparison between real world stacking and the proposed system was also conducted to test plausibility of the behavior.

1.1 Initial Problem

Is it possible to simulate stacking using a different approach than most state-of-the-art work?

1.2 Stacking in Children's Play and Video Games

An important part of the development of a child is a type of play called constructive play [Fromberg & Bergen, 2006]. Constructive play at first focuses on arranging objects in patterns, and later on it focuses on creating structures and systems. In constructive play children learn about physics, object balance and material properties. Another important element is that the child sets his or her own goals, and then attempts to reach them. This type of play is important as most other types of play after the age of 7 years depend on constructive play. As children use electronic devices ever more, as well as spending ever more play time with digital games, it would be beneficial if some of these learning elements can be incorporated into digital games. Allowing a child to set his or her own goals is not always supported in video games, which are often goal driven; however, this is mostly a question of game design and as such will not be covered in this project. As properties of materials are mostly of tactile domain, this element of constructive play will also not be investigated. The most interesting elements are physics on small object scales, and object balance. It should be possible to recreate these

elements with current technology. Enabling small object stacking behavior can also give game designers more freedom to allow for child specified goals.

1.3 Stacking in Simulation

The technical motivation for the project has several aspects. The first and most basic motivation is being able to simulate stacking at small object scale. Stacking in itself is difficult to simulate but it has been done by academic systems as seen in Section **Error! Reference source not found.**. It is however unclear if these systems also work on small object scales.

One of the two main motivations for the project is to investigate whether using a corrective system on top of an existing physics engine is a viable method to simulate scenarios that the underlying engine cannot simulate. Currently, implementing a system to handle a new type of simulation scenario requires modifying or even completely rewriting large parts of the physics engine. This makes it complicated to change the feature set of physics engines during a production. Being able to use modules that run on top of the underlying physics engine allows for more flexibility of systems, and to potentially save resources by only using calculations required for specific simulation scenarios.

The other main motivation is to investigate the feasibility of using a different approach to simulate stacking. In the literature, the problem of stacking is usually solved by improving the precision of the physics system to the point that a large number of objects can be interacting with each other in a stable manner. The presented project takes a different approach to solve the problem of stacking than most literature, but the approach is similar to [Hsu & Keyser, 2012]. Instead of calculating the forces acting on each individual object, properties of the stack are calculated and the overall stack stability is continuously evaluated. While the objects are part of the stack, their movements are restricted. When the system detects an unstable stack, the unstable part is removed from the stack and the object's movement is no longer restricted. It is then expected that the underlying physics engine handles the collapse of the structure.

1.4 Research Question

Is it possible to simulate user-relevant stacking scenarios, in a visually plausible manner, using stack behavior instead of behavior of individual objects in the stack, and implement the system on top of a closed source physics engine?

Chapter 2 Previous Work

This chapter reviews current approaches to solve stacking in physics engines. Several papers have presented solutions for solving this problem by developing better solvers for physics engines and other methods.

2.1 The Normal Structure of a Physics Engine

A normal rigid body physics engine can be divided into 2 main phases. The first phase is the collision detection system. This system is further split into two parts. Broad phase collision detection simplifies the problem by using very efficient methods to find potential collisions, for instance finding bounding box overlap. The narrow phase collision detection is then used to define if the collision meshes are actually touching or overlapping. The second part is a collision response system. This is where contact forces are calculated. A simple approach is an impulse based collision response. Velocity changes for the colliding objects are modeled using impulses that match the time step of the physics engine. The future position of an object is therefore expressed as differential equations that can be solved using numerical methods or similar solvers. Most stacking based research is based on optimizing the collision response system, often by using more advanced response paradigms than the impulse based response, or by improving the solvers. [Eberly, 2004]

2.2 Literature Related to Object Stacking

A widely cited paper on stacking objects in a physics engine is [Erleben, 2007]. The approach presented by the paper uses a standard collision detection algorithm, but presents its own complementary formulation, solver and error correction algorithms. The paper first derives the complementarity formulation used by the solver. It then presents different base frameworks for solvers and presents its own variation on one. The method proposed in the paper is velocity based as opposed to the conventional impulse based approach. The paper discusses an error correction scheme that is partially based on stack layers derived from the contacts of objects. The method proposed by Erleben is compared to several other physics engines, and performs better on all stacking scenarios while still running at near real-time speeds. The paper shows that using the structure of a stack has to some extend been considered before and is a viable approach.

A slightly more recent paper [Kaufman, Sueda, James, & Pai, 2008] focuses on accurately simulating friction and enabling friction dependent behavior. The method proposed in the paper is usable in both rigid and deformable bodies' simulation. Although not the main focus, this system can simulate stacking scenarios that are friction dependent. The stacking scenarios only maintain a stable structure, not constructing it during simulation. The system performs at near real-time speeds. This paper demonstrates that simulating correct friction behavior

supports a system's ability to simulate stacking. The paper also presents an interesting simulation scenario in its card house stack. This scenario is mainly dependent on friction to remain stable.

[Hsu & Keyser, 2010] describe an approach to improve the performance of random stacks in simulation. The contribution of this paper is an advanced object sleep paradigm. By forcing object sleep, depending on pile specific conditions, more realistic pile behavior is achieved. The algorithm runs at real time and can even provide slight performance improvements. Although the authors have implemented the algorithm inside a physics engine, this approach should be able to function independent of the underlying physics engine. The approach is similar to the one describe in this project. The system proposed by Hsu and Keyser is, however, only meant to simulate random stacks.

A later more recent paper by the same authors [Hsu & Keyser, 2012] presents a method to induce object sleep based on local equilibrium. The algorithm periodically revises the grouping, and removes objects from groups if objects are no longer in equilibrium or are impacted by fast objects. The system is created to allow for art-directed stacking, and not necessarily realistic stacking; however, the system cannot create stacks that are massively unstable. The system is tested on both stack and random stack scenarios. The system is able to maintain stacks created before simulation, but it is not clear if, though highly likely that, the system can handle stacks that are created during simulation. Only random stacks are shown to be created during simulation. The system runs at real-time to near real-time speed. The system seems to be a modification of the Bullet physics library but the algorithm can be used as an extension. This system is the closest prior work to what is proposed in the presented project. The system proposed by [Hsu & Keyser] uses an open source engine, so the authors still have access to the source code even though it is claimed to work as a general extension. The method presented by this project uses a closed source engine; thus, only API calls are possible. The system described by [Hsu & Keyser] uses grouping of objects as the main method of generating stable stacks whereas the method proposed by the presented project uses object sleep state to create stacks. The focus of the paper is artistic control over stacks, whereas the focus of the presented project is stack simulations under user control.

A paper by [Han, Hsu, McNamara, & Keyser, 2013] sets out to test two commonly used hypotheses in physics animation. The first is that users are unable to perceive distortions in the simulation that come from approximated simulation methods. The second hypothesis is that freezing transformations of objects in a random pile does not affect the visual plausibility of a simulation. Both hypotheses are confirmed in a user study. The paper identifies 4 variables that may have an effect on the validity of the hypotheses but leave it up to future works to find the extend of this effect.

This paper represents one of the rare times where alternate simulation methods are tested on users for visual plausibility. Doing such user tests is a good way of confirming or disproving the presumed visual plausibility of a developed system.

The paper [Sakurai & Miyata, 2013] proposes a method for arranging objects in predefined shapes and piles. The presented method first distributes objects over the desired shape so that they are in contact with nearby objects. It then refines the distribution by moving each object to minimize the overlap with other objects, or remove the object if overlap cannot be reduced sufficiently. The method is evaluated by expert users who found that the system functions well. The method does not conduct simulation. This paper focuses on the generation of object piles and which elements make a visually plausible pile. Its approach goes from the overall shape of the pile on to the state of the individual object in the pile. Although a different method and goal is used the general idea of basing the method in the overall shape is similar to this project.

A paper from the industry is [Tonge, Benevolenski, & Voroshilov, 2012]. It presents a method for impulse based rigid body simulation optimized for parallel computing. The paper splits object mass to match the number of contacts with other objects. This enables parallel computation of impulse calculation without imposing any serialization on the algorithm. The proposed method reduces simulation jitter and yields similar performance to conventional methods. The method has worse convergence than one of the standard methods, but it manifests itself as increased contact compliance instead of jitter. The method proposed by the paper is able to simulate a card house stack although with visual artifacts in the form of bouncing. This paper focuses on utilizing modern computer architecture to increase performance. The method is the only one of the cited papers that is designed for parallel computing. It reduces simulation jitter, which is stated to be a problem for visual plausibility and stack behavior, and thus indirectly improves stacking behavior. As this method is a modification of a physics engine, this type of approach is not possible to implement with an extension system as is the goal of this project.

Chapter 3 METHOD

This chapter describes the method and the general implementation of the method. It is split into five parts. The first two parts describe the problems with the base physics engine, and determine the scope of the implementation. The last three parts describe the proposed solution without going into the actual code.

3.1 PROBLEMS WITH BASE PHYSICS ENGINE

Physics engines for games generally are optimized for fast performance and low calculation time, and are often modifications of the impulse based system briefly described in Section 2.1 . The physics engine used for this project is the implementation of Nvidia Physx that is built into the Unity 3D game engine [Unity Technologies, 2014]. This physics engine is a closed source engine, but a paper from the company [Tonge, Benevolenski, & Voroshilov, 2012] suggests that the engine uses some variation of the impulse based simulation method.

By observing stacking behavior it can be seen that the physics engine first operates normally for an impulse based system. It detects collision points and handles the object contacts as expected. However, when the stack becomes large, the associated forces also grow and the physics engine changes behavior and no longer consistently detects collision points. It is assumed that the engine is using penalty forces instead of contact response methods. This leads to more unstable structures, and stability errors where bricks that should be stable move or rotate, leading to collapse of the stack, see [Video on the enclosed DVD "4stack Unity Raw"] . This type of error is the main problem preventing stacking with this physics engine.

Another problem that appears with larger stacks of objects is residual energy in the stack. Calculation inaccuracies and/or micro collisions cause motion to be inducted into the object stack and the stack begins to sway visibly on its lateral axis. This is mostly a visual artifact as stability is not much affected. However, it makes adding to the pile more difficult and for structures that have limited stability it may prove problematic.

The physics engine is designed for low levels of gravity. With low gravity and mass settings, the engine is able to handle simple stacking fairly well. However, when simulating at smaller scale, where the gravity acceleration relative to object size is larger, the engine starts showing the problems described above. One case, where the engine is not able to handle stacking, is small object scale with realistic gravity. Here, the problems are very pronounced, and collision detection stops after only a few objects stacked on top each other, see [Video on the enclosed DVD "4stack Unity Raw"]. As this scale is the primary scale used for constructive play, stacking support on this scale is important for the simulation of constructive play with small wooden bricks for children.

3.2 Scope Limitation

In order to limit the scope of the implementation a focus on a specific simulation task is selected. For this project, simulation of wooden bricks is chosen to demonstrate the algorithm. The bricks are a type of wooden bricks called KAPLA [KAPLA, 2014]. These bricks are wooden boxes where the sides have a ratio of 1:3:15. They are made from untreated pinewood which provides a high level of friction. Some of the bricks can be seen in Figure 1. The convex nature of the bricks helps to keep the initial implementation simple, as cases of multiple contact areas between two bricks are not possible. The bricks also have in good approximation a uniform mass distribution across the object. Basing the system in actual physical objects allows for comparisons between the simulations and reality to investigate the accuracy of the simulations.

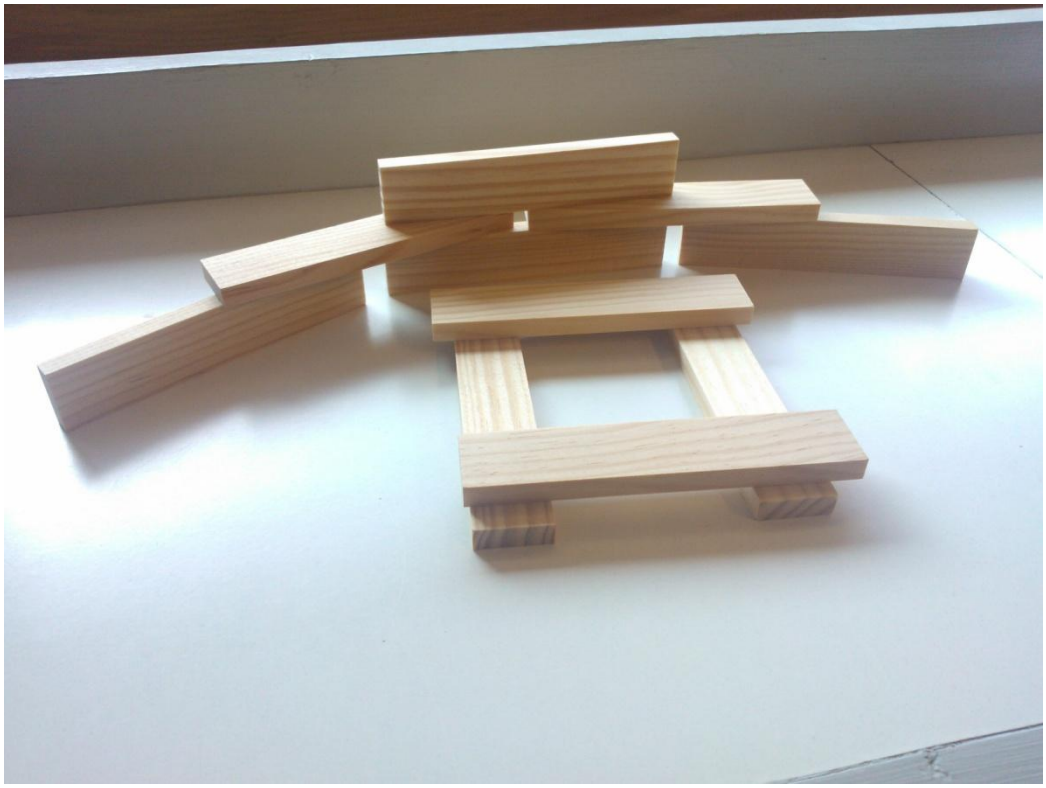

Figure 1: Several KAPLA bricks placed in simple stacks.

3.3 Brick Stability System - Basic Version

This section describes the simple version of the brick stability system. This is the base model and the method that takes care of the most seen stacking scenarios. For a discussion on user stacking types, see [Portfolio Chapter 4]. The general idea of this method is to evaluate stability based on the centre of mass of objects and contact areas between different objects. The system calculates the centre of mass for the structure above a brick, and uses this to determine the stability of the structure.

3.3.1 General Algorithm

The brick stability system for simulating stacks of objects is based on the centre of mass and the contact area. If the projection of the centre of mass of an object is outside the contact that supports the object, the object is going to tilt and fall off its support as shown in Figure 2. As the method is an extension of a physics engine, objects will start under the control of the physics engine. If an object is detected to be in a stable configuration, the brick stability system will freeze it in the physics engine and take over stability calculations. If unstable configurations are detected, the brick stability system will wake up the brick and the physics engine will then simulate the brick until it again returns to a stable configuration.

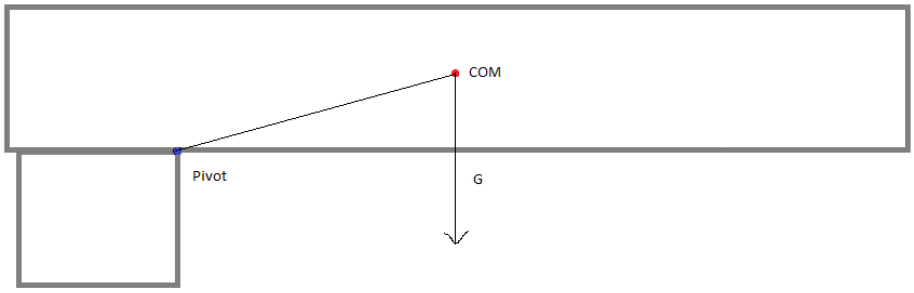

Figure 2: Example of an unstable configurations. Under gravity the top object will pivot over the edge of the contact and subsequently fall off. COM notes the object centre of mass and G is the gravity vector.

For every frame, new contacts must be found and any potential new additions to the stack must be tested, the state of the stack must be updated and the overall stability of the stack must be tested. First, all contacts are added to the data structure of the system, then all contacts between two objects that are both in unstable positions are removed as they cannot be resting on a part of the stack. Contacts coming from impacts above a certain speed are also disregarded. Contact areas between stable and potentially stable objects are then calculated based on the remaining contact points. Using these contact areas potential bricks are tested for stability, and if they are in a stable configuration, they are added to the stack. The system then updates stability data through the stack. Lastly, the stack is tested for overall stability. This is done by calculating the centre of mass for everything above each object, and testing if the vertical projection of this centre of mass is outside of the main contact area of the object. If it is then the object and everything above it is woken up and flagged as unstable.

For each object, a series of information is needed for the algorithm. This information is stored in a data structure as seen on Figure 3. Every object that is

part of the stack stability system has a brick data object associated with it. If the object is part of the environment, it is flagged as a ground object.

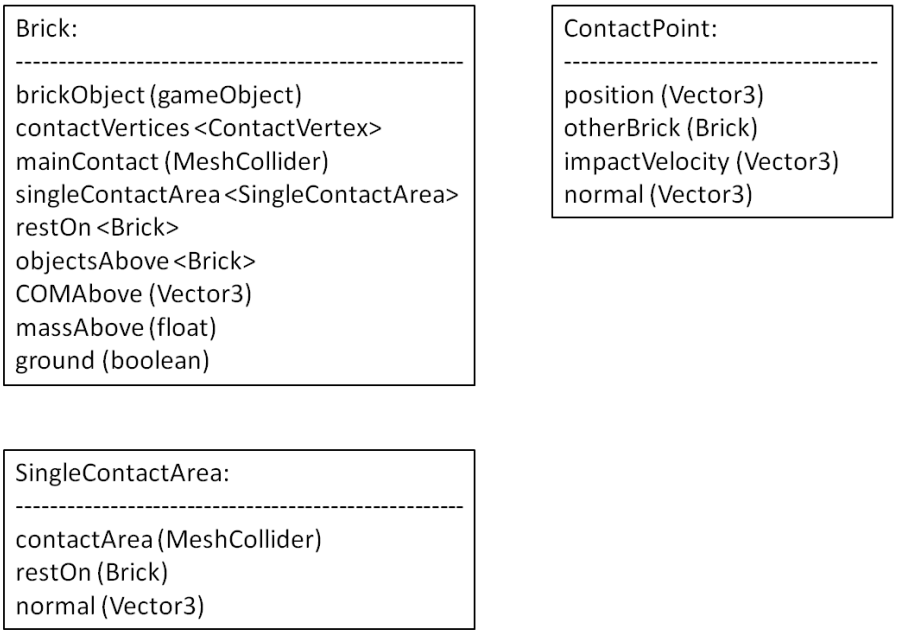

Figure 3: Datastructure for each object. <type> denotes that it is a list of type and (type) indicates that it is an instance of type.

3.3.2 Assumptions

The system assumes that bricks in stable contact with other bricks in the stack have infinite friction between them. This is justified as common stacking behavior is usually not based on high friction. It is assumed that a stable stack cannot start on top of unstable bricks and that bricks in the stack cannot move to a different position without becoming unstable. This is a conservative approach, so if the proposed system is in doubt it refers to the physics engine. If a brick is in a position where its entire weight can be supported by a single brick, even though it rests on more, the entire weight is supported by this brick. This is based on common stacking behavior, as there are often central elements in user structures . If a brick rests on more than two bricks, the weight is distributed equally between these unless the previous case is true. This assumption is supported by the stacking behavior observed in the preliminary test [Portfolio Chapter 4]. If a brick is moving and rotating above a defined speed, it is assumed that the brick cannot be in a stable position. This threshold takes micro collisions of the game engine into account.

3.3.3 Contact Area

For the algorithm to function, the contact areas between bricks must be known. This contact area is a central part of the algorithm. Two types of contact areas are needed. The first type is a series of contact areas between a brick and each of the bricks upon which it rests. The second type of contact area is the convex hull of all the contact areas the brick rests on. The base physics engine makes contact points available for each contact between two objects. In order to take advantage of built in functions, in particular for ray casting, however, it is preferable to have contact areas as meshes.

In order to calculate the convex hull of the contact points, the contact points first have to be arranged in order around the periphery of the contact area. To do this, a gift wrapping algorithm is used. The used algorithm is based on a 2D implementation from [Abdulin, 2012]. As the contact points are points in 3d, modifications have to be made to the algorithm to make sure that a valid result is returned. First the algorithm finds the starting point which is the contact point with the lowest x coordinate. It marks it as the first point on the periphery and sets this as last visited. It then takes a random point and sets it as the test point. The algorithm goes through all contact points and tests if the point is to the left of the line from the last visited point to the test point, this is done on a projection on the y=0 plane. If a point is to the left of this line this point is set as the test point. After all points have been iterated, the algorithm adds the test point to the periphery and sets this point as last visited. The algorithm continues this approach until the point found as last visited is the same as the starting point. At this time all points on the periphery have been visited. Lastly, the list of original periphery points are converted to a mesh, and assigned to the brick as a mesh collider in order to take advantage of built in functions. The mesh is not planar, but this does not pose a problem, as the algorithm does not need planar contact areas to function correctly.

If the number of contact points is either 2 or 3, new points are added to make it possible to generate a mesh. In the case of two contact points, two new points are generated a short distance away from the original points in the direction perpendicular to the two original points. The direction is parallel to the x and z axis (y being the vertical axis). If only one point is given, three new points will be generated spaced out along the x and z axis and the original point will be moved accordingly to create a diamond shaped contact mesh with the centre at the original contact point.

This algorithm is used to generate both the contact area for resting on individual bricks, but also for the combined resting area. In the last case, the contact points from everything rested on is used as input. The algorithm returns the convex periphery, which is used as the combined contact area.

3.3.4 Single Object Stability

In order to test the stability of a single brick, the system tests if the centre of mass is above the contact area. This can be done fairly easily by a ray cast from the objects centre of mass straight down. If this ray cast hits the convex hull of the contact areas, then the brick is considered to be in a stable configuration.

When a brick is found to be in a stable configuration, it is added to the stack. References between the objects in contact with each other are made, and the brick is marked as stable. This can be seen in Figure 4. The simulation of the brick in the physics engine is stopped and the brick acts as an immovable object in order to still be active in the collision detection.

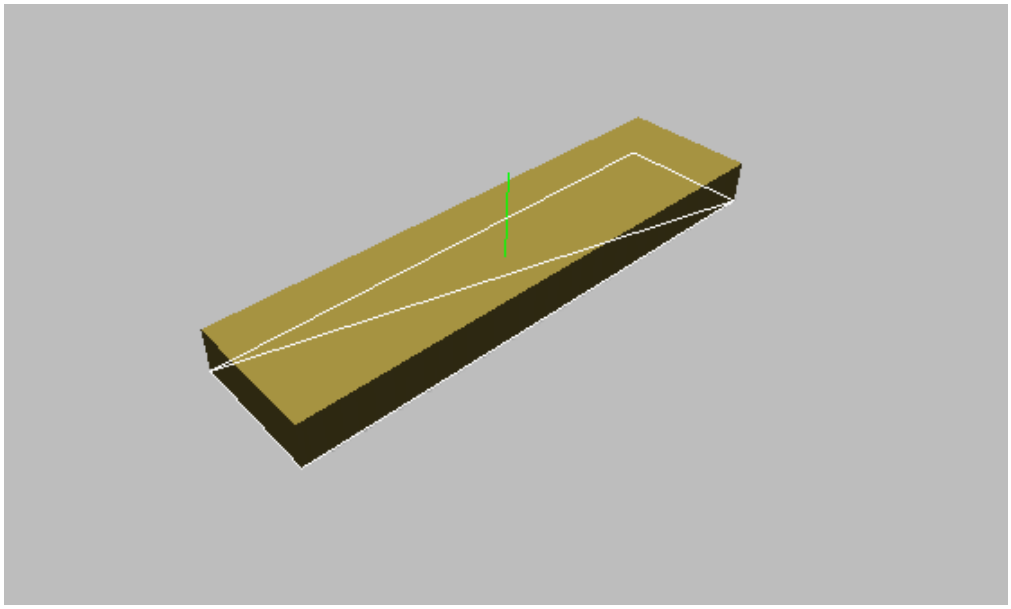

Figure 4: A single stable brick. The green line is a ray marking the centre of mass, and the white mesh is the main contact area.

3.3.5 Single Branch Stability

The simplest type of stacks is single branch stacks. In this type of stacks, objects only rest on one other object. In order to be able to calculate the stability of the entire stack, the centre of mass of the stack above a brick must be calculated. This calculation is done when the brick status is being updated. If the brick is not part of the stack, its centre of mass (COM) is updated from the information in the physics engine. The centre of mass for objects above and including the brick itself (COM-above) is set to the same value, as the COM-above is including the brick itself, and there is nothing above the brick as it is not part of the stack.

If the brick is part of a stack, the COM-above is calculated using Equation 1 where COM is the COM-above, and Mass is the mass above. The subscript denotes which brick the COM and mass belongs to: Brick means the brick in question and brick above means the value for the brick above.

$$COM_{brick} = \frac{(COM_{brick} * Mass_{brick}) + (COM_{brick\ above} * Mass_{brick\ above})}{Mass_{brick} + Mass_{brick\ above}} \quad [1]$$

This averages the centre of mass according to the mass of the brick itself and the mass of everything above. Before these calculations begin, the COM above and mass above are initialized to the values of the brick itself. The mass above is summed afterwards according to Equation 2.

$$Mass_{brick} = Mass_{brick} + Mass_{brick\ above} \quad [2]$$

The Centre of mass above can be seen illustrated in Figure 5.

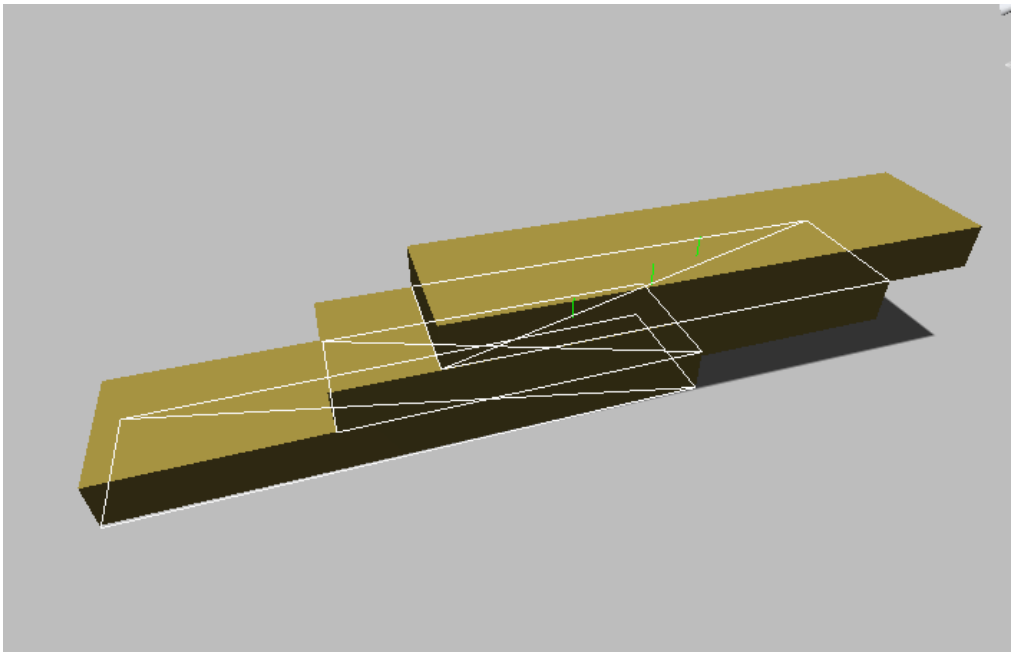

Figure 5: A single branch stack. Note the COM above in green, and the contact area in white. As the COM above is for the stack above a brick, the top matches the brick's own COM whereas the lowest COM matches the centre of mass of the entire stack.

After this update, the stability of the stack is then calculated using the same method as for single object stability, just using the COM above for the test. If the COM-above is outside of the object's main contact, the brick and all bricks above it are removed from the stack and flagged as unstable and returned to be simulated by the physics engine.

3.3.6 Multi-Branch Stacking

There are two different types of branching that may happen in the stack. The first is splitting branching, where two or more bricks rest on a single brick. The other type of branching is merging branching, where one brick rests on two or more bricks. An example of the two branching types is seen in Figure 6.

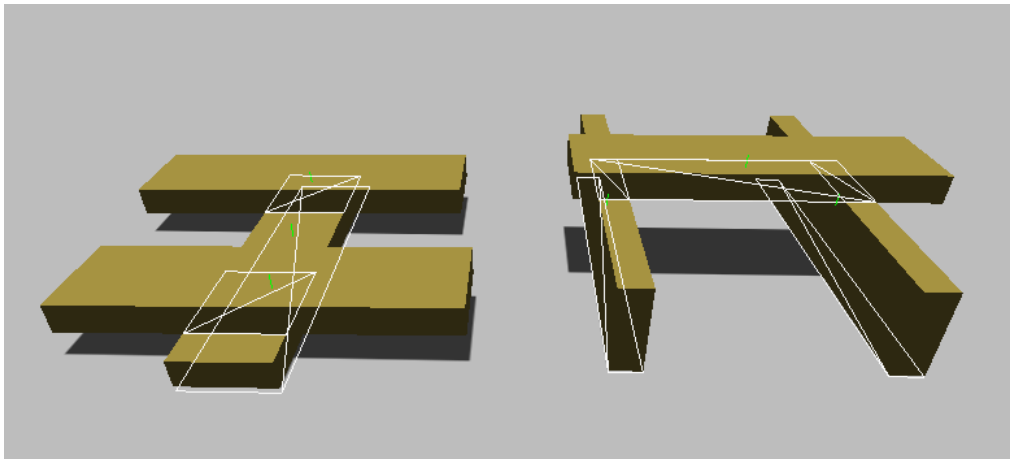

FIGURE 6: THE TWO CASES FOR MULTI-BRANCHING. TO THE LEFT IS SPLIT BRANCHING WHERE TWO BRICKS REST ON ONE. TO THE RIGHT MERGING BRANCHING WHERE ONE BRICK RESTS ON TWO.

The split branching is the simplest version of branching to account for. This is done by using the same method as for single branching, but repeating it for every brick above. In an implementation, these are identical as the calculations will be in a loop structure that will only iterate once in the non branching case.

The second type of branching is calculated differently. Before calculating split branching, it is tested if any brick above is resting on two or more bricks. If so, the system is dealing with merging branching. For a more clear explanation, the brick above the one being updated will be denoted brick A. For merging branching there are three different cases. The first case is that the entire weight of the stack and brick A can be supported by a single brick. This is true if the COM-above of brick A is within one of the individual contact areas. This is tested similar to the single bricks stability, but using the individual contact areas instead of the combined. If this is true, the entire weight of brick A will be on this supporting brick. If the supporting brick is the brick being updated, it will be updated similar to the single case. If it is not, no update of data will be made (if there are also other bricks resting on the brick these will update normally). The second case is that brick A is resting on more than two bricks. In this case the update is done similar to the single branch case but the mass of brick A for the calculations is reduced to a fraction matching the number of bricks it rests on.

The last case is that brick A is resting on two bricks. In this case, the distribution between the two supports is calculated. The calculation is based on the setup seen in Figure 7.The assumption is that gravity would cause a torque on the pivot point on the other support brick as if the supporting brick that is being updated was not present. The equivalent force of this torque would be exerted on the supporting brick that is being updated and can be recalculated as a weight. This assumption goes both ways, and generates the distribution of mass.

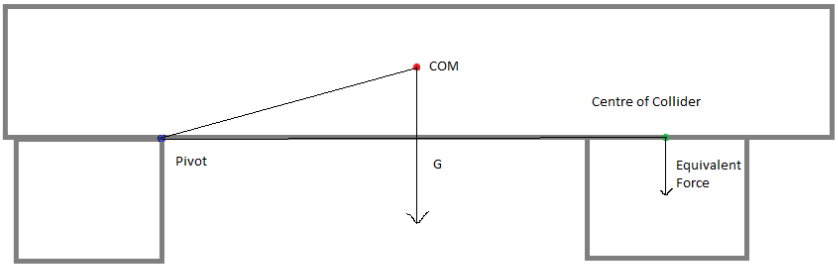

FIGURE 7: THE 2 BRICK MERGING-BRANCHING CASE. THE BOTTOM LEFT BRICK IS BEING UPDATED.

The pivot point is the contact point closest to the centre of mass. The torque exerted on this point by gravity is calculated using the equation for torque [Eberly, 2004] seen in Equation 3, and with the implementation based values in Equation 4.

$$Torque = r \times F \quad [3]$$

Where r is the lever arm vector and F is the force on the lever arm.

$$Torque = (COM_{above} - pivot) \times gravity \quad [4]$$

From the torque a force on the other contact area can be calculated. Using Equation 5, and rearranging it to Equation 6, the downwards force equating the torque can be calculated, and the equation used in the implementation is seen in Equation 7.

$$|Torque| = |r||F| \sin\theta \quad [5]$$

Where |r| is the magnitude of the lever arm, |F| is the magnitude of the force and θ is the angle between the lever arm and the force vector.

$$|F| = \frac{|Torque|}{|r| \sin\theta} \quad [6]$$

$$|Force| = \frac{|Torque|}{|Centre\ of\ contact\ area - pivot| * \sin\theta} \quad [7]$$

Where θ is the angle between the line from the pivot to the centre of the contact area and gravity vector.

The magnitude of the force is found, and it is parallel to the gravity vector. The weight equivalent is calculated using the equation for the force of gravity seen in Equation 8 and as implemented in Equation 9

$$F = mg \quad [8]$$

$$Equivalent\ mass = \frac{|Force|}{|g|} \quad [9]$$

Since the force and gravity vectors are parallel their magnitudes can be used instead.

Finally the brick can then be updated using the same base formula as in the single brick case but with the equivalent mass substituted as seen in Equations 10 and 11.

$$COM_{brick} = \frac{(COM_{brick} * Mass_{brick}) + (COM_{brick\ above} * Equivalent\ mass)}{Mass_{brick} + Equivalent\ mass} \quad [10]$$

$$Mass_{brick} = Mass_{brick} + Equivalent\ mass \quad [11]$$

3.3.7 Wake Up Function

The wake up function is the part of the system that handles returning a brick from being part of the stack back to the physics engine. The function is called any time a stable brick is identified as being in an unstable configuration. The function first flags the brick as being active in the physics engine's simulation again. Then it calls itself on any brick that rests on the brick being woken up. This ensures that bricks cannot be resting on nothing. After this the different data structures associated with the brick will be returned to the start configuration, and some garbage collection will be run, to remove the meshes for the contact areas from memory. Any objects upon which this brick is resting will also have the brick removed from the array of objects resting on them. A similar system exist in [Hsu & Keyser, 2010] where it is used to wakeup parts of their piles; however, the objects that are affected are determined by their position in the pile, and not by their relations to other bricks, apart from this the systems are similar.

3.4 Brick Stability Leaning Version

The base version of the brick stability system, can handle most stacking scenarios observed in the preliminary test [Portfolio Chapter 4]. One type of stacking scenario that the base version cannot handle is leaning bricks. Here the simple system cannot accurately simulate the bricks leaning against each other. Due to the requirement in the base version that objects can only enter the stack if they are resting stably on the stack bricks, leaning against each other in stable configuration cannot enter the stack as neither of them is already in the stack.

3.4.1 Assumptions

The system for leaning stability uses a few different assumptions than the base version. One major difference is that friction is used as part of the stability calculations and is not assumed to be infinite. The algorithm assumes that the brick is leaning or unstable, as any bricks otherwise stable would be found so by the base version. For friction calculations, static friction is assumed.

3.4.2 Contact Area

The contact areas for this improvement are generated in a similar way as the contact areas for the base version; however, if only two or less points are given, the generated points are not spread out horizontally, but perpendicular to the normal vector of the contact point. These contacts are handled in a different list than the contacts from the base system, because some contact points required for the leaning version's contact points may be discarded for the base version. The contact areas used by the leaning version also have a normal vector associated with them, as these contact areas are not parallel with the x-z plane.

3.4.3 Stability Calculations

The algorithm for stability uses some of the same principles as the weight distribution function in the base version, but for the leaning stability friction calculations are included as well. First, the system identifies the number of objects in leaning contact. If the number is one or less, leaning stability is not possible. If the number is over 2 the system also returns false, as this scenario is too complicated for the current model to handle; however, basic stacking is tested first, so if a brick can be stable using the basic version more than two contact bricks are possible. The system handles the case where a brick is in contact with two other bricks (or ground object), and at least one of these are part of a stack. This case is the one most commonly seen in the preliminary test [Portfolio Chapter 4].

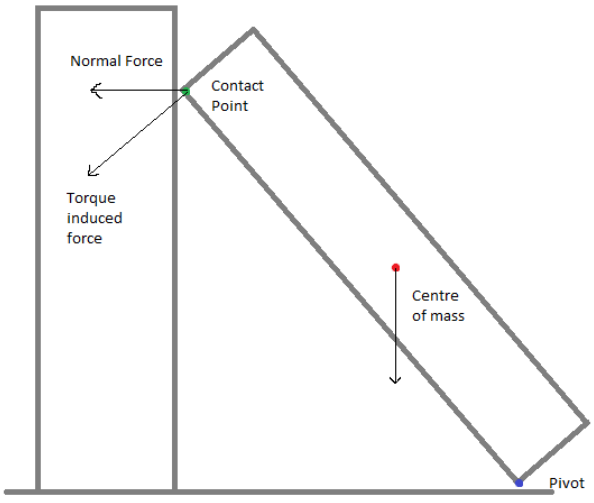

Figure 8: The leaning case. The two unstable objects and a ground plane. The pivot point (blue) and contact point(green) is shown on the sketch along with the normal force and the torque induced force.

The first step is to identify the lowest contact area (blue dot in Figure 8). The next step is to find the contact point, on the x-z plane, closest to the centre of mass of the brick. This point will function as the pivot point. The torque from gravity is calculated in the same way that it is calculated in the weight distribution function in the base version with Equation 4. To find the direction in which the torque acts on the second contact (green in Figure 8), the cross product between the torque vector and the line from the pivot to the contact point is normalized. Using an approach similar to the one in the base version Equation 7 the magnitude of this torque-induced force can now be found. By multiplying the magnitude of the torque induced force with the force vector found before, the force vector is found. In order to calculate the amount of friction supporting the brick from the contact point, the force perpendicular to the contact surface must be calculated. This force is found by projecting the force vector onto the inverted normal of the contact area. Using this normal force, the Coulomb friction can be calculated using Equation 12 [Erleben, Sporring, Henriksen, & Dohlmann, 2005].

$$\left|F_f\right| = u|F_n| \qquad [12]$$

Where $|F_f|$ is the magnitude of the friction force and F_n is the normal force.

Next the sliding force, the force the brick wants to move with, in the same direction as the friction force, must be calculated in order to find the direction for the friction force. Normally this would be done with a projection, but as the normal force has already been calculated, and the sliding force can be found by subtracting the normal force from the force the brick is generating on the contact.

Now the force on the pivot point can be calculated using Equation 13. The normal force is inverted as this is the force the other brick is assumed to be exerting in a stable configuration to avoid object penetration. It is assumed that the friction at the contact point holds as much weight of the brick as the friction force allows.

$$F_p = -F_n + g * mass - F_f \qquad [13]$$

Where F_p is the total force on the pivot point, F_n is the normal force on the contact point, and F_f is the friction force on the contact point.

Then the normal force for the pivot point can be found the same way as before, as well as the friction and sliding force. If the friction force is larger than the sliding force, then the brick is considered stable.

3.5 Integration with Base Version

In order for the system to function alongside the basic version, some changes to the framework have to be done. In order to avoid conflicts between the two types of contact areas, they are kept in different lists and function independently. At first the contact points for the leaning stability version are calculated, then the contact

points that should not be used for the basic version's contact points are removed, and the basic versions contact areas are calculated. In terms of checking for stability, bricks are first tested for stability using the basic version, and after this using the leaning version if the basic version does not find the bricks stable. In order to make sure that bricks that are leaning are not entered into the basic version's stable state, if the bricks have two contacts, the height difference between the two contact areas are tested to be less than a threshold height. Additions to the wake up function are also made to account for the extra contact areas.

3.6 Collision Handling

The system proposed in this project does not focus on collision behavior. The system implemented is therefore a fairly simple system. Significant improvements can be made to this system following several different approaches as discussed in [0].

3.6.1 Assumptions

For the implementation of collision handling, it is assumed that the force of impact to the structure only occurs from the impacting object. If multiple objects impact a structure, only the direct impacts will be accounted for, and no force from second impact objects will be used. The impact behavior will be threshold based: If the impact energy is lower than the threshold nothing will happen and if it is higher the impacted part of the structure will wake up and the impact will be handled by the physics engine.

3.6.2 Calculations

The impact handling tests if the impact energy is higher than the potential energy of the brick impacted and the structure resting on it. If the brick associated with a contact point has a speed above the rest threshold, the kinetic energy is calculated based on the speed of the impacting brick and it's mass. After this the potential energy of the impacted brick and the stack is calculated based on the bricks centre of mass for itself and the stack above it, and the centre point of the bricks main contact area. If the kinetic energy is higher than the potential energy, the brick and the stack above is woken up. If the potential energy is highest, the stack remains stable and the physics engine handles the impact as if the stack did not move.

Chapter 4 CAPABILITIES TEST

4.1 DESIGN

This test is designed to investigate the capabilities of the proposed system compared to the unmodified underlying physics system and a modified version of the underlying physics system which is optimized to simulate the stacking scenarios. The details of these two systems can be found in [Portfolio 5.3]. The test is composed of a few stacking scenarios which are based on common stacking behavior observed in a preliminary test [Portfolio Chapter 4]. In addition to this, scenarios containing impacts from other bricks are included to show the system's response to impacts. The simulations are recorded using screen capture. The simulations are compared to each other and recordings of the performance of real life bricks in similar stacking configurations. The simulation stacks are constructed during simulation, with one brick being added to the stack at around 1 second intervals until the structure is completed or collapses. The scenarios that will be used are described below.

4 stack: A structure where two parallel bricks are placed on each layer, and rotated 90 deg for the next layer. This structure is very stable and is mostly limited by construction accuracy and the quality of the base, that the tower rests on.

Card house: A card house like stack, consisting of inverted V structures made from two bricks which support a level brick that rests on the peaks of two V's. On top of the level brick is another inverted V. In the end the structure resembles a pyramid see Figure 10 in the next chapter, the stack in the capabilities test is one level larger.

Slant tower: A structure similar to 4 stack except that each layer is offset from the center in the same direction making the structure appear to lean. The structure will collapse at a certain height.

4 stack impact: The same structure as 4 stack is created, and subsequently hit laterally by a brick in the top of the structure. Another test is done where the impact is in the 3rd layer of the tower. The tower being impacted will be 13 layers tall.

4.2 RESULTS

The results of testing the 4 stack structure showed that the proposed system is capable of generating towers over 25 levels high. The base physics system is only capable of getting this structure to a height of four levels high. The structure begins to sway after two layers are added due to what is assumed to be micro collisions, and the swaying increases as more layers are added. The collapse of the structure starts as the bricks in the lowest layer tilt, generating lateral motion as well as letting the whole structure start an almost free fall. The system with modified

parameters fared somewhat better but collapsed in a similar manner as the base system when it reaches a height of around 15 layers tall.

The card house stack can be simulated by the proposed system and behaves stable. The unmodified base physics system is not able to generate a stable inverted V. The modified system is able to generate a stable lower level of the card house stack; however, when subsequent levels are added, the structure shows signs of beginning the same swaying as seen in the 4 stack, except that the unstable nature of the structure makes structural collapse almost immediate.

From this point on the unmodified base system will be considered as not being able to simulate the structures being tested.

For the slant tower test, the proposed system makes the tower collapse when the structure reaches 9 levels high. The collapse seems to happen at the correct time, the collapse of the tower, however, is too vertical. In reality the structure is going to pivot on the lowest level bricks, and remain almost intact until the structure hits the ground. The modified base physics system collapses at a height of 10 levels. The behavior of the collapse itself follows reality closely, pivoting on the lowest level and remaining mostly intact before impact with the ground.

For the top impact, the proposed system removed the brick at the point of impact from the stack, and the brick falls over on top of the structure. However, in some simulations the stack was unaffected by the impact and in some subsequent impacts led to partial structure collapse. The parameter modified base system suffered complete structure collapse following the impact at the top. This seems to be due to a lateral momentum being imparted on the structure by the impact. Impacting at the top of a real version of the tower at different velocities, results in either the collapse of the top one or two layers, or a partial structure collapse.

When the impact is in the bottom of the structure, the proposed system does not react to the impact. This is due to the more threshold-like nature of impact handling with this system. The impacting brick looses all lateral velocity and falls down next to the tower. Increasing the impact velocity did not produce a different result, and using very high velocities causes errors with the collision detection system. For the optimized base system, the impact energy is imparted to the bricks in the 2nd to 4th layers, mostly the 3rd layer, causing a large deformation of the tower. This deformation causes instability of the structure and subsequent structure collapse. Conducting this experiment with physical bricks has different results depending on the impact speed. At lower speeds only small deformations are observed in the tower, as impact speed increases the deformations in the tower becomes larger. The deformations are mostly isolated to directly hit bricks. With impacts at high speed, higher than in the simulation, the deformations cause structural collapse in the real bricks tower.

4.3 Discussion

The results of the capabilities test show that the proposed system is able to simulate stack stability in a plausible way. The system is capable of simulating stable structures and detects when a structure exceeds its stability. The point of collapse for the proposed system is close to reality, collapse behavior however is not always realistic. This is due to the fact that the underlying physics system handles the collapse itself, and the underlying system may not be able to correctly simulate the partial stabilities that may happen during collapse. Unmodified, the underlying physics system is incapable of simulating any of the structures in the test. A modified version is to some extend able to simulate all of the scenarios, with the loss of realistic constants like gravity and friction constants. The difference in impact behavior is expected as the impact handling scheme in the proposed system is very simplified.

Chapter 5 Test 2 User Test

5.1 Design

A user test was conducted to test how users perceive the proposed system in comparison to a conventional system [Portfolio 5.3]. Naive to intermediate knowledge users are used to evaluate the system. These users are recruited from medialogy students on 4th semester and higher. These test participants are expected to have a novice to intermediate understanding of physics systems as well as intermediate knowledge of non standard user interfaces. The last fact helps as the test systems are controlled by a 6 DOF controller [Portfolio 5.2 6DOF Controller]. The test compares two different physics systems. One is the simulation system proposed in the report, and the other is the physics system in Unity 3D, with tweaked constants to allow for stable stacking. The participants are testing two stacking types. The first is a leaning structure as seen in Figure 9. Participants continue to build it until it falls over, ensuring that each participant experiences a structure collapse. The second structure resembles a card house Figure 10 and consists mostly of slanted bricks. As this structure is not inherently unstable, participants will add bricks flat on top until a collapse occurs.

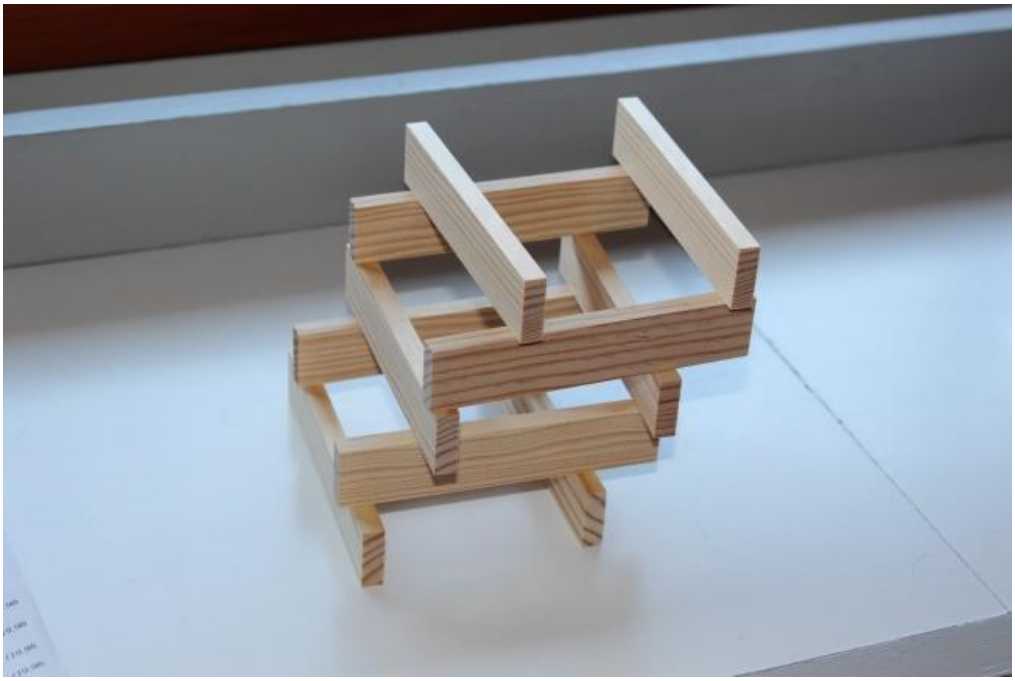

Figure 9: The Leaning structure, only the first four layers are shown.

FIGURE 10: THE CARD HOUSE STRUCTURE.

The test procedure starts with the participant using physical bricks to construct a pre-designed structure type until it collapses. This enables the participant to learn how the physical bricks act when constructing the structure and how they act during the structure collapse. After constructing with the physical bricks, the participant is introduced to the interface of the test program and introduced to the controls. The participant is asked to do a short training session consisting of docking tasks. The participant is then asked to construct the structure in the two systems that are presented in a counterbalanced order. After this first part, the participant is asked to fill out a questionnaire [see DVD in the "test" folder] investigating the perceived realism of the two systems, as well as the ease of the using the systems. The process is repeated for the second stack type.

The test is filmed and screen capture is used to record the program during participant interaction. The video recordings were used in cases of interesting interactions using the physical bricks. The screen recordings were used to investigate problems with the algorithm that come up during the test.

5.2 RESULTS

20 people participated in the test, and a single person participated in a pilot test. As slight modifications were made after the pilot test, data from the pilot test are not used. Of the 20 people that participated, all 20 completed the first part of the test, 3 participants, however, were not able to stack the bricks in the second stacking scenario, and as a result of this only 17 people completed the second part of the test. This fact also caused the counterbalancing for the second part of the test to not be valid. The answers from the questionnaire are analyzed and the results are

presented in Figure 11 and Figure 12. The questions are answered on a Likert scale, with 1 being strong preference for the base physics system with modified parameters and 5 being strong preference for the presented stacking system.

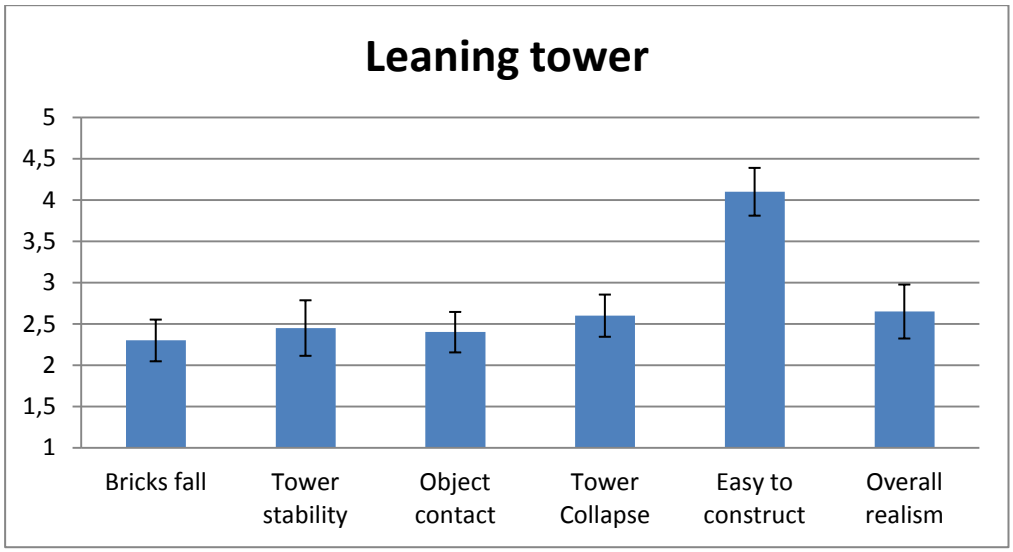

FIGURE 11: PARTICIPANT REPORTED REALISM FOR LEANING TOWER STACK. 1 REPRESENTS THE BASE PHYSICS SYSTEM MOST REALISTIC, AND 5 REPRESENTS PROPOSED SYSTEM MOST REALISTIC. 1 COLUMN FOR EACH QUESTION WITH ERROR BARS FOR STANDARD ERROR.

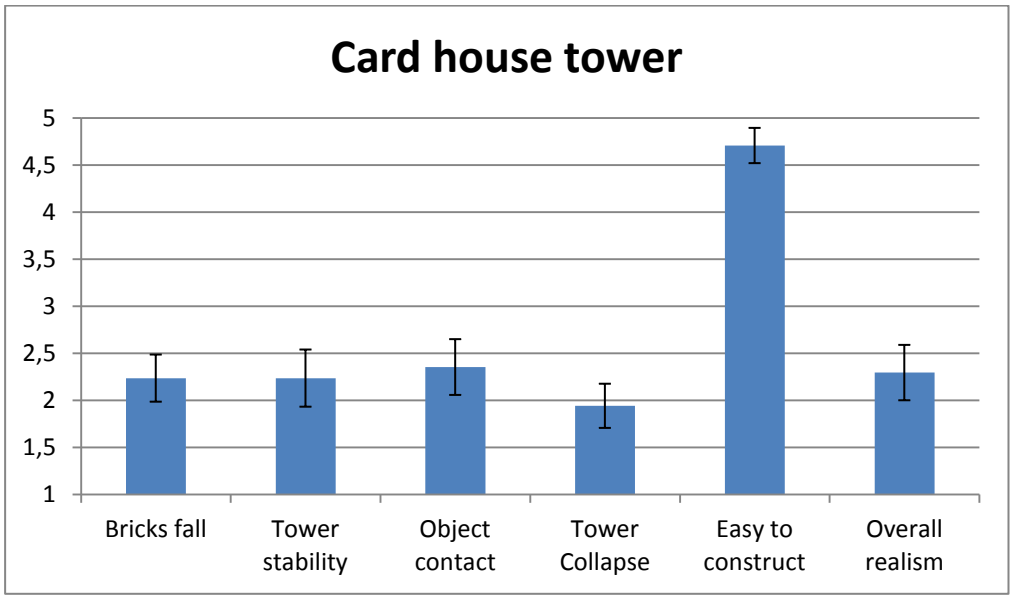

FIGURE 12: PARTICIPANT REPORTED REALISM FOR CARD HOUSE STACK. 1 REPRESENTS THE BASE PHYSICS SYSTEM MOST REALISTIC, AND 5 REPRESENTS PROPOSED SYSTEM MOST REALISTIC. 1 COLUMN FOR EACH QUESTION WITH ERROR BARS FOR STANDARD ERROR.

As can be seen from the graphs, the participants preferred the base physics system as the overall most realistic version and the most realistic on all of the elements of realism investigated. They preferred the proposed system for ease of construction.

The comments from the questionnaire have been analyzed and the general tendencies will be described below. 12 participants commented that the proposed system seemed the most stable of the two, however, more participants, 8, saw this as less realistic. One of the 4 participants who found the proposed system more realistic was the only one to reference the real bricks. 3 participants noted that the gravity in the proposed system was higher than in the base physics system, and 1 participant felt the gravity was too high and commented on it as unrealistic. The second main point from the comments is the impact handling which 10 participants commented on. Several elements are commented on. Clipping with the ground was commented on by 4 participants. Collisions with the selected brick in the base physics version was commented on by 4 participants all finding it realistic. The limited impact effect in the proposed system was found realistic by 2 and unrealistic by 3 participants. 2 participants commented on latency in the proposed system and found it unrealistic. 3 participants comment on the shaking that occurs in both versions when the structure is near or beginning to collapse. 1 participant found it unrealistic, whereas 2 found it realistic.

In the card house test, participants mostly commented on the stability, unrealistic structures, and the sense of sticky bricks and partial collapse of structures. 7 participants felt that the tower using the proposed system was more stable than the tower using the base physics system, all of them felt that the added stability was unrealistic. 3 participants noticed structures they did not expect to be able to stand in the proposed system, see Figure 13 and Figure 14 for examples. 2 participants commented that the bricks in the proposed system felt sticky in regards to realism. 2 participants also commented that partial structure collapse felt unrealistic.

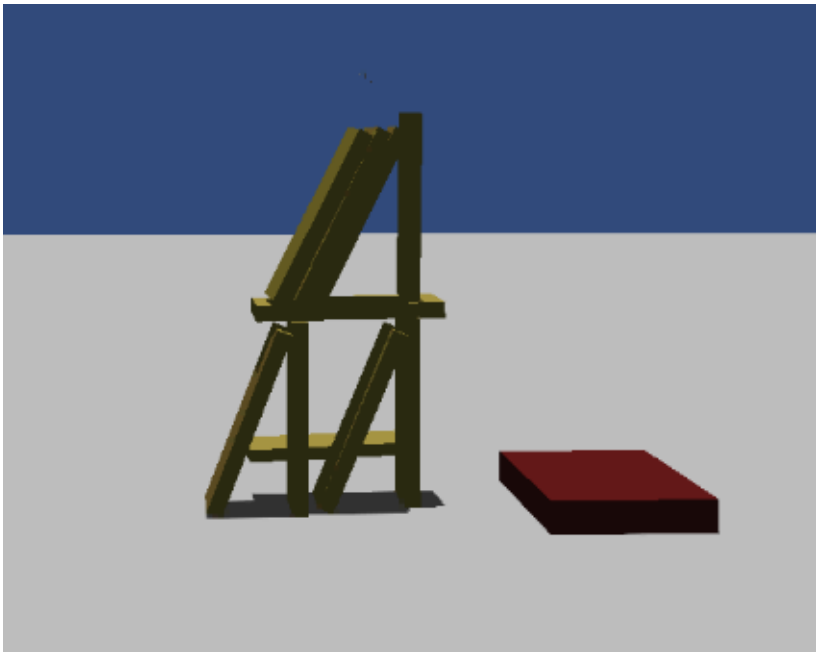

FIGURE 13: THE THREE LEFT BRICKS ON TOP SHOULD MAKE THE TOP RIGHT BRICK FALL OVER. THIS DOES NOT HAPPEN DUE TO THE SIMPLIFICATIONS OF THE MODEL.

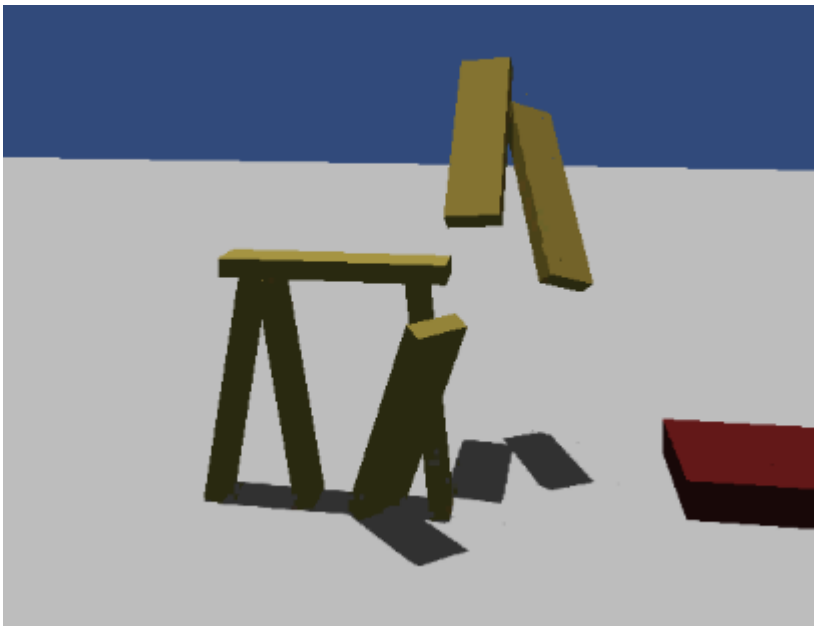

FIGURE 14: THE BOTTOM BRICK LEANING AWAY FROM THE STRUCTURE LEANS ON THE RIGHT MOST BRICK BUT IS NOT IN A STABLE CONFIGURATION BUT THE SYSTEM ASSUMES IT IS SUPPORTED. THE TWO "FLOATING" BRICKS ARE UNDER USER CONTROL.

5.3 Discussion

When looking at the results of the user test, a trend can be seen in that the participants generally saw the base physics system as the most realistic. The participants, however, were more in favor of the proposed system in terms of ease of use. Looking at the comments this can be explained by the fact that the proposed system is more resistant to accidentally hitting the tower with the bricks being moved. The brick that is being moved has its position and rotation determined by the controller. Due to the way the selection is programmed the base physics system register zero impact velocity. As a result of this it most likely uses penalty forces to repel the impacted brick; this system seems to cause large repulsion forces when interacting with immovable objects. As the proposed system does not use penalty forces, or have an error correction system, impacting bricks with a selected brick have no effect except in cases of massive object intersection which were not seen in the test. As the proposed system mostly ignores impacts from the controlled brick the structure is less likely to fall over due to accidental impacts from the user controlled brick as it is placed.

Participants recognized the proposed system as the more stable version. The discrepancy as to its realism is to some extend explained by the differences between the two versions. The proposed system does not allow for any movement of the bricks that are part of the structure while the structure is stable and not interacted with. This is realistic due to the aforementioned lack of interactions. However, it may seem unrealistic when placing new bricks or hitting the structure. Conversely, the base physics system reacts to these impacts but when the structure is near instability or if it is particularly large, small swaying motions can be seen in the structure even when left alone. The fact that participants considered brick shaking before collapse to be realistic could be explained by referring to the real bricks. When participants were about to place a brick that would topple the structure, many of them would shortly lift the brick again and let the structure return to a stable state. Only after doing this a few times would they let go of the brick and let the structure collapse.

For the card house structure participants were unanimous in finding the increased stability to be unrealistic. This differentiation can be explained mostly from the different structure. As the card house structure is much more unstable in nature, any sway in the base physics system will cause structure collapse, thus the participants do not see the swaying problems mentioned above. The more rigid and interaction-proof proposed system will seem overly stable as any mistaken interaction should collapse the structure. Participants reporting sticky bricks can be explained by the simplifications in the way the proposed system handles leaning bricks, which can lead to false positives, this also explains some of the unrealistic configurations. Partial structure collapse can be seen with the proposed system due

to two factors: One is the simple impact handling, if only parts of the structure become unstable and are removed from the stack, the rest of the structure will be more resistant to collapse from the impacts of the collapsing part. The other is the threshold nature of adding moving bricks to structures. If the entire structure collapses, but part of it does not gain significant velocity, it is possible for these bricks to reform part of the structure shortly after collapse. Partial structure collapse has been observed with the physical bricks during the test.

Chapter 6 Discussion

From the two tests that have been conducted it can be seen that there are some discrepancies between real world comparisons and user perception of visual plausibility. This discrepancy may be partially explained by the more random nature of the interactions in the user test compared to the scripted nature of the capabilities test. Where the users generate less ideal configurations that may be more difficult to simulate for the model than the more calculated object configurations from the scripts in the capabilities test.

6.1 Limitations to the Proposed System

The proposed system has several issues that need to be addressed. The major ones are the interaction system and the leaning brick model. It is clear that the interaction model had a large part in the results of the user test. The very unresponsive nature of the interaction model of the proposed system was commented by a large number of participants. Similar to this, the very rigid stability of the proposed system felt unrealistic to the users. From these general observations it seems that interactivity and semi-stability are more important aspects of stable stacking than first assumed. It may also be the case that stack deformations is an important aspect of stable stacking. The interaction scenario from the capabilities test was the only scenario that didn't have realistic behavior, in that no deformations of the stack were observed.

The leaning brick model should also be revised. It cannot simulate configurations where more than two contacts are present. In order to simulate this, a more advanced method is required. The current method also generates false positives for stability which must be corrected. It is possible that a different approach to leaning stability is required.

Adding bricks in the middle of the stack is not observed in user stacking, but it is none the less a feature the proposed system lacks. The system is not able to add a brick, that other bricks already in the stack rests on, to the stack.

Chapter 7 Conclusion

An extension for a physics engine is proposed and implemented. A technical test has been performed and it found that the system performed better than the normal physics engine with realistic settings, and that a modified version of the physics engine performed only slightly worse than the proposed system when compared to real bricks. A user test showed that users found the physics engine with modified parameters more visually plausible than the proposed system, but preferred the proposed system in terms of ease of construction. The research question for this project was:

Is it possible to simulate user-relevant stacking scenarios, in a visually plausible manner, using stack behavior instead of behavior of individual objects in the stack, and implement the system on top of a closed source physics engine?

The proposed system was able to replicate realistic behavior in the proposed simulation scenarios with exceptions of the interaction scenario. The physics engine alone, could not handle the scenarios when using realistic settings. With modified parameters, it was able to a lesser degree to simulate the scenarios. For simulating user-relevant stacking scenarios the hypothesis is partially confirmed with the exception of interaction.

In terms of user-perceived visual plausibility, the hypothesis is rejected as users preferred the standard system. In terms of perceived realism users significantly favored the physics engine with modified settings over the proposed system. It is expected that the simplicity of the interaction system is a part of this result.

Chapter 8 FUTURE WORKS

Further work on the proposed system can focus on two clear improvements. The first is a more responsive collision handling system. Enabling this should improve the perception of users, and add functionality to the model that is not sufficient in the current version. The second area of improvement is the leaning stability algorithm. Improving the model to be more robust would be a significant improvement, as in its current form it can misidentify unstable configurations as stable and has limited functionality for advanced configurations. An additional improvement would be to allow the system to add bricks in the middle of a stack.

Work could be done to investigate which if any academic physics engines can simulate small scale object as physics engines are not normally shown working under these scenarios. A study investigating users' perception of realistic settings for small object simulation could also be an interesting topic, as some users found the (realistic) gravity too high.

APPENDIX A

BIBLIOGRAPHY

Abdulin, P. (2012). Retrieved May 21, 2014, from https://stackoverflow.com/questions/10020949/gift-wrapping-algorithm

Eberly, D. H. (2004). *Game Physics.* Morgan Kaufmann.

Erleben, K. (2007). Velocity-Based Shock Propagation for Multibody Dynamics Animation. *ACM Transactions on Graphics , 26* (2), 20.

Erleben, K., Sporring, J., Henriksen, K., & Dohlmann, H. (2005). *Physics-Based Animation.* Charles River Media .

Fromberg, D. P., & Bergen, D. (2006). *Play From Birth to Twleve: Contexts, Perspectives, and Meanings* (2nd ed.). Routledge Taylor & Francis Group.

Han, D., Hsu, S.-W., McNamara, A., & Keyser, J. (2013). Believability in Simplifications of Large Scale Physically Based Simulation. *Proceedings of the ACM Symposium on Applied Perception* (pp. 99-106). ACM New York.

Hsu, S.-W., & Keyser, J. (2012). Automated Constraint Placement to Maintain Pile Shape. *ACM Transactions on Graphics , 31* (6).

Hsu, S.-W., & Keyser, J. (2010). Piles of Objects. *ACM SIGGRAPH Asia 2010 papers.* ACM NewYork.

KAPLA. (2014, May 21). Retrieved from http://www.kapla.com/kapla/accueil.en.htm

Kaufman, D. M., Sueda, S., James, D. L., & Pai, D. K. (2008). Staggered Projections for Frictional Contact in Multibody Systems. *ACM SIGGRAPH Asia 2008 papers.* ACM New York.

Sakurai, K., & Miyata, K. (2013). Modelling of Non-Periodic Aggregates having a Pile Structure. *Computer Graphics Forum , 33* (1), 190 - 198.

Tonge, R., Benevolenski, F., & Voroshilov, A. (2012). Mass Splitting for Jitter-Free Parallel Rigid Body Simulation. *31 (4)* . ACM New York.

Unity Technologies. (2014, May 21). Retrieved from https://unity3d.com/unity/quality/physics