Simulating Small-Scale Object Stacking Using Stack Stability

Kronborg Thomsen, Kasper; Kraus, Martin

Published in:
Poster Paper Proceedings of the 23rd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2015)

Publication date:
2015

Document Version
Early version, also known as pre-print

Simulating Small-Scale Object Stacking Using Stack Stability

Kasper Kronborg Thomsen
Aalborg University
Department AD:MT
Rendsburggade 14
Denmark, 9000 Aalborg
kasperkronborgthomsen@gmail.com

Martin Kraus
Aalborg University
Department AD:MT
Rendsburggade 14
Denmark, 9000 Aalborg
martin@create.aau.dk

ABSTRACT

This paper presents an extension system to a closed-source, real-time physics engine for improving structured stacking behavior with small-scale objects such as wooden toy bricks. The proposed system was implemented and evaluated. The tests showed that the system is able to simulate several common stacking scenarios, which the base physics engine cannot simulate.

Keywords

Simulation, object stacking, engine extension, game physics.

1 INTRODUCTION

The real-time simulation of dense structure stacking in game physics engines is a challenging research topic with many applications in video games and virtual reality simulators. In this paper, stacking refers to dense structured stacking as seen when creating deliberate structures. Our main focus is on the simulation of stacking of small-scale objects, which is particularly challenging because gravitational acceleration is relatively large compared to the size of these objects. To achieve a real-time simulation of such stacking scenarios, we propose a correction system for a closed-source physics engine, namely the Nvidia physics engine that is integrated in the Unity game engine.

2 PREVIOUS WORK

A widely cited paper on stacking objects in a physics engine was presented by Erleben [Erleben, 2007]. His approach uses a standard collision detection algorithm but includes its own complementarity formulation, solver and error correction algorithms. 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 was compared to several other physics engines and performs better on all stacking scenarios while still running at near real-time speeds. His publication showed that using the structure of a stack has to some extend been considered before and is a viable approach.

A slightly more recent paper by Kaufman et al. [Kaufman et al., 2008] focuses on accurately simulating friction and enabling friction-dependent behavior. Their method is usable in both rigid and deformable bodies' simulation. Although not the main focus, their system can simulate stacking scenarios that are friction-dependent in near real time. However, it does not construct stable structures during simulation. This paper demonstrates that simulating correct friction behavior supports a system's ability to simulate more stacking scenarios, in particular a cardhouse stack. This scenario is interesting because it is mainly dependent on friction to remain stable.

Hsu and Keyser [Hsu and 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 objects into an object sleep state, 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 independently of the underlying physics engine. The approach is similar to the one describe in the present paper. The system proposed by Hsu and Keyser is, however, only meant to simulate random stacks.

A more recent paper by the same authors [Hsu and Keyser, 2012] presents a method to sta-

belise stacks by adding constraints between objects 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 designed to allow for art-directed stacking, but not necessarily realistic stacking; however, the system cannot create stacks that are massively unstable. It was tested on both structured stack and random stack scenarios. The system is able to maintain stacks created before simulation, but it is not clear if the system can handle stacks that are created during simulation as this scenario was not shown. The system runs at real-time to near real-time speed. It is the closest prior work to what is proposed in the present paper. However, the system uses the Bullet physics library, an open source engine, thus the authors have access to the source code even though it is claimed to work as a general extension. The method presented by the present paper uses a closed-source engine; thus, only API calls are possible. The system described by Hsu and Keyser [Hsu and Keyser, 2012] uses grouping of objects as the main method of generating stable stacks whereas the method proposed by the present paper uses object sleep state to create stacks. The focus of Hsu and Keyser [Hsu and Keyser, 2012] is artistic control over stacks, whereas the focus of the present paper is stack simulations under user control.

A paper by Han et al. [Han et al., 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 are due to 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. This paper represents one of the rare instances where alternate simulation methods are tested on users for visual plausibility.

One common feature of all the papers proposing new simulation methods is that these are not demonstrated to work on small object scale while our system is designed specifically for small scales as they are characteristic for toy bricks.

3 METHOD

As the proposed 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 object stability system will freeze it in the physics engine and take over stability calculations. If unstable configurations are detected, the object stability system will wake up the object and the physics engine will then simulate the object until it again returns to a stable configuration.

For every frame, new contacts must be found and any potential new additions to the stack of frozen objects must be tested, the state of the stack must be updated and the overall stability of the stack must be tested. Firstly, 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 objects 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 all objects 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.

The contact area between two objects is found by generating the convex hull of the projections of the contact points between the two objects onto the ground plane. To test the stability of a single object, a vertical ray is cast from the centre of mass of the object. If the ray hits the contact area, the object is considered stable.

For more than one object stacked above another object, different types of stack branching have to be considered. The simplest form of branching is a single branch, where objects are stacked on top of each other one at a time, thus only one object rests on another object. To calculate the stability of this stack, objects above a given object must be taken into consideration when calculating stability. To this end, the centre of mass of an object and all objects stacked on top of it is defined as C_{stack}, which is calculated using Equation 1 where C_{obj} is the centre of mass for the object, m_{obj} is the mass of the object, $C_{\text{stack,above}}$ is C_{stack} for the stack above the current object and $m_{\text{stack,above}}$ is the mass of the stack above the current object. Analogously, m_{stack} is the mass of the stack calculated by Equation 2.

$$C_{\text{stack}} = \frac{\left(C_{\text{obj}} m_{\text{obj}}\right) + \left(C_{\text{stack,above}} m_{\text{stack,above}}\right)}{m_{\text{obj}} + m_{\text{stack,above}}} \tag{1}$$

$$m_{\text{stack}} = m_{\text{obj}} + m_{\text{stack,above}} \tag{2}$$

Two types of multi-branch stacking are distinguished. One is split branching, where multiple objects rest on one object. This case is handled similarly to the single branch case, except that Equations 1 and 2 are used for every object resting on the current object. The second type of branching is merge branching. In this case an object rests on more than one object. For a more clear explanation, the object above the one being updated will be denoted object A. For merge branching

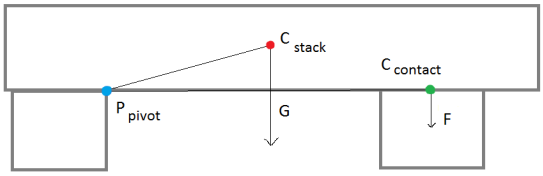

Figure 1: The two object merge-branching case. The bottom right object is being updated.

there are three different cases. The first case is that the entire weight of the stack and object A can be supported by a single object. This is true if the C_{stack} of object A is within one of the individual contact areas. This is tested similarly to the single object stability, but using the individual contact areas instead of the combined contact area. If this is true, the entire weight of object A will be on this supporting object. If the supporting object is the object being updated, it will be updated similarly to the single case. If it is not, no update of data will be made (if there are also other objects resting on the object, these will update normally). The second case is that object A is resting on more than two objects. In this case the update is done similarly to the single branch case but the mass m_{stack} of object A for the calculations is reduced to a fraction matching the number of objects it rests on.

The last case is that object A is resting on two objects. In this case, the distribution between the two supports is calculated. The calculation is based on the setup depicted in Figure 1. The assumption is that gravity would cause a torque around the pivot point at the other support object as if the supporting object that is being updated was not present. The equivalent force of this torque would be exerted on the supporting object that is being updated and can be recalculated as a weight. This assumption goes both ways, and generates the distribution of mass.

The pivot point is the contact point closest to C_{stack}. The torque exerted on this point by gravity is calculated using Equation 3.

$$\mathbf{T} = \left(C_{\text{stack}} - P_{\text{pivot}}\right) \times \mathbf{G} \tag{3}$$

Where $\mathbf{T}$ is the torque, P_{pivot} is the pivot point, and $\mathbf{G}$ is the gravitational force.

From the torque a force $\mathbf{F}$ on the other contact area can be calculated. Using Equation 4, and rearranging it, the downwards force equating the torque can be calculated, and the equation used in the implementation is shown in Equation 5.

$$|\mathbf{T}| = \left|C_{\text{contact}} - P_{\text{pivot}}\right| |\mathbf{F}| \sin\theta \tag{4}$$

Where θ is the angle between the lever arm and the force.

$$|\mathbf{F}| = \frac{|\mathbf{T}|}{\left|C_{\text{contact}} - P_{\text{pivot}}\right| \sin\theta} \tag{5}$$

The direction of the force is parallel to the gravity vector. The equivalent mass, m_{equiv}, is calculated using the equation for the force of gravity, i.e., $m_{\text{equiv}} = |\mathbf{F}|/g$.

The object can then be updated using the same base formula as in the single object case Equation 1 but using the equivalent mass instead of the $m_{\text{stack,above}}$.

The algorithm for leaning stability uses some of the same principles as the weight distribution function seen before, but 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 and this case was not observed in the test how users stack with toy bricks. Flat stacking is tested first, so if a object can be stable using the above version, more than two contact bricks are possible. The system handles the case where an object is in contact with two other bricks (or ground objects), and at least one of these is part of a stack.

First the lowest contact area (blue dot in Figure 2) is found. The next step is to find the contact point, on the horizontal plane, closest to the centre of mass of the object. 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 previously described with Equation 3. To find the direction in which the torque acts on the second contact (green in Figure 2), 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 Equation 5 the magnitude of this torque-induced force can now be computed. By multiplying the magnitude of the torque induced force with the force vector found before, the force vector is determined. In order to calculate the amount of friction supporting the object 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 6 [Erleben et al., 2005].

$$|\mathbf{F}_{\text{friction}}| = \mu \, |\mathbf{F}_{\text{normal}}| \tag{6}$$

Next the sliding force, the force the object 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. Usually this would be done with a projection, but as the normal force has already been calculated, the sliding force can be found by subtracting the normal force from the force the object is generating on the contact. Now the force on the pivot point can

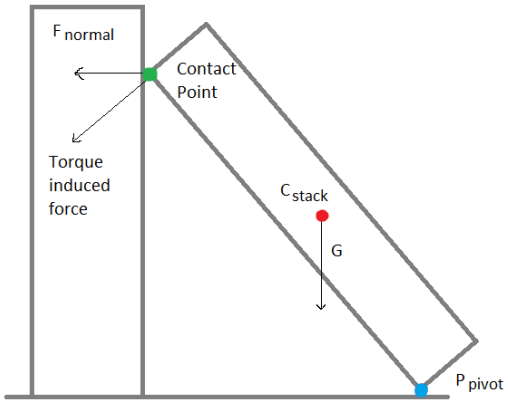

Figure 2: Leaning object

be calculated using Equation 7. The normal force is inverted as this is the force the other object 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 object as the friction force allows.

$$\mathbf{F}_{\text{pivot}} = -\mathbf{F}_{\text{normal}} + \mathbf{G} - \mathbf{F}_{\text{friction}} \tag{7}$$

Where $\mathbf{F}_{\text{pivot}}$ is the total force on the pivot point, $\mathbf{F}_{\text{normal}}$ is the normal force on the contact point and $\mathbf{F}_{\text{friction}}$ is the friction force on the contact point. $\mathbf{G}$ is the gravitational force on the pivot point.

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

4 RESULTS

The tests were designed to investigate the capabilities of the proposed system compared to the base physics system. The tests included three stacking scenarios (see Figure 3) which are based on common stacking behavior observed in a preliminary user test. The simulations were compared to each other and to recordings of the behaviour of real bricks in similar stacking configurations. The simulation stacks were constructed during the simulation, with one brick being added to the stack at approximately 1 second intervals until the structure is completed or collapses. For better comparison with real-world objects, KAPLA wooden bricks were used as the base for all scenarios.

Figure 3: The three stack types, from left to right: 4-stack, card house and leaning tower.

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 card house stack can be simulated by the proposed system and behaves stable. The base physics system is not able to generate a stable inverted V which is the base of the structure.

As the leaning tower scenario is to some extend similar to the 4-stack, the base physics engine is not able to simulate the scenario. The proposed system makes the tower collapse when the structure reaches 9 levels. The collapse appears to occur at the correct time, however, it is too vertical since real towers tend to pivot on the lowest level and remain mostly intact before impact with the ground.

5 CONCLUSION

We presented an extension system to a physics engine, which is able to simulate common stacking scenarios that the base physics engine cannot simulate. These results encourage further research on improvements of the proposed system, in particular related to the simulation of leaning bricks, which is currently not robust and can in some cases allow incorrect stability.

6 REFERENCES

[Erleben, 2007] Erleben, K. (2007). Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2):article no. 12.

[Erleben et al., 2005] Erleben, K., Sporring, J., Henriksen, K., and Dohlmann, H. (2005). *Physics-Based Animation*. Charles River Media.

[Han et al., 2013] Han, D., Hsu, S.-W., McNamara, A., and Keyser, J. (2013). Believability in simplifications of large scale physically based simulation. *ACM Symposium on Applied Perception*, pages 99 – 106.

[Hsu and Keyser, 2010] Hsu, S.-W. and Keyser, J. (2010). Piles of objects. *Proc. SIGGRAPH Asia*, 29(6):article no. 155.

[Hsu and Keyser, 2012] Hsu, S.-W. and Keyser, J. (2012). Automated constraint placement to maintain pile shape. *ACM Trans. Graph.*, 31(6):article no. 150.

[Kaufman et al., 2008] Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K. (2008). Staggered projections for frictional contact in multibody systems. *Proc. SIGGRAPH Asia*, 27(5):article no. 164.