EUROPEAN CONSORTIUM FOR MATHEMATICS IN INDUSTRY

ECMI Modelling Week 2018
University of Novi Sad, Faculty of Sciences

STORING YOUR RANDOM OBJECTS

Thomas Götz (University of Koblenz and Landau)

Jacob Jon Hansen (Technical University of Denmark)
Desiré Nilsson (Lund University)
Ivana Gengeljacki (University of Novi Sad)
Kirill Kiselev (Saint Petersburg Polytechnic University)
Monika Žunji (University of Novi Sad)
Sampsa Kiiskinen (University of Jyväskylä)

2018-07-15 – 2018-07-21

Contents

1 Introduction

The problem with storing random objects is encountered already in childhood as when cleaning one would have to put building blocks of different sizes and shapes in a box for storing. If individually organized, little space will be wasted in the container and the lid of the box will most likely easily close. But saving time, most would not organize the blocks but just throw them in, and this poses an interesting problem, will the blocks fit inside the container so that the lid will close properly? To study this question one would like to know what the packing density of the container is when the content is put in randomly.

The packing density is likely dependant on a variety of properties. Among other things the shape of the object put in it, the shape of the container and the sizes of the object, both in comparison to each other and to the container. When all objects are completely round is a well studied problem, therefore this report aims to study other shapes. To simplify the problem, only objects of the same shape is considered, cubes. Further simplification is reducing from 3D to 2D, leaving the objects to be squares. The assumption that the container will not move is made. When packing a box of randomly oriented objects that does not really fit, one would probably shake it to rearrange the configuration. This changes the problem and add complexity, it will therefor not be addressed.

Originating from the solution with balls the biggest difference is that the orientation of the ball when it falls into the container does not affect the result. But for cubes, the orientation in all three dimensions matters, as ignoring it will lead to physically impossible configurations.

An analytic solution exists for packing spheres in a cubical container Visscher and Bolsterli 1972, however there is no obvious analytic solution to the problem of packing squares/cubes when thrown into a container at random. It is however possible to use mathematical modelling to describe the dynamics of such an event and thus the approach in this work is to use simulation to try to answer the problem.

Additionally the following assumptions are made in this work:

- We assume that we have infinitely many cubes / squares to our disposal, such that we can always try to throw in one more.
- We consider the problem of packing objects into a container which should have a lid. Thus if at the end of a simulation an object is not fully within the container, we will have to remove it to be able to close the lid.

Since our work is mainly simulations, a video is available showing all the simulations running in real time: https://youtu.be/MUkOYQRp8Nw.

Code is available at: https://github.com/jakejhansen/ECMI2018

2 Report content

Section 2.1 explains our overall procedure in a pen-and-paper fashion. Section 2.2 explain our actual implementation of three different simulation environments. Section 2.3 deals with the results of our experiments and the report ends with a conclusion followed by a brief section on group work dynamics and an appendix on order metrics.

2.1 Bottom-to-Top Reconstruction

We decided to start with a bottom-to-top reconstruction algorithm used for granular dynamics (Pöschel and Schwager 2005), but specialize it for rectangular particles. It works as follows.

First square will land on the bottom of the container. If landing flat on one side, it is in a stable condition and the square is fixed at its position. If at all tilted when landing, only one corner of the square is supported which is not a stable state. It will therefor tilt in the direction of the center of mass, in the unlikely event that the center of mass is exactly above the supported corner, the direction doesn't matter. It will continue to tilt until it is stable which could happens in two ways. Either it falls on its side or it hits the wall of the container and stays tilted, supported by one point on the bottom and one on the container wall. This is also a stable state and it will thereafter stay fixed.

Following squares have the opportunity to interact with each other. Assuming a setup like the top left in Figure 1 we will clarify the process. Another square, number six, is added to the box, it falls until in contact with either the bottom of the container of another square. In this case it is another square, number five, and as it is tilted it is not in a stable state and will therefore rotate in the direction of the center of mass. If the center of mass is inside the two connection point between the two squares five and six, all is well. As this is not the case, the square will rotate around the second point of contact until it again reaches a second connection point. This results in the square not being supported at all and it will fall straight down until it again has a connection point. In this case, there is two as square number six falls flat onto square number two. Center of mass is outside the connection points and the square will tilt around the connection point closest to the center of mass until reaching the next connection point, the corner of square one. The center of mass is now between the connection points, and the square is finally at rest.

The procedure continues until squares can no longer fit into the container anymore. This is said to be when the corner of any square is outside the container. The packing density is then simply the area of the squares divided by the total area of the container.

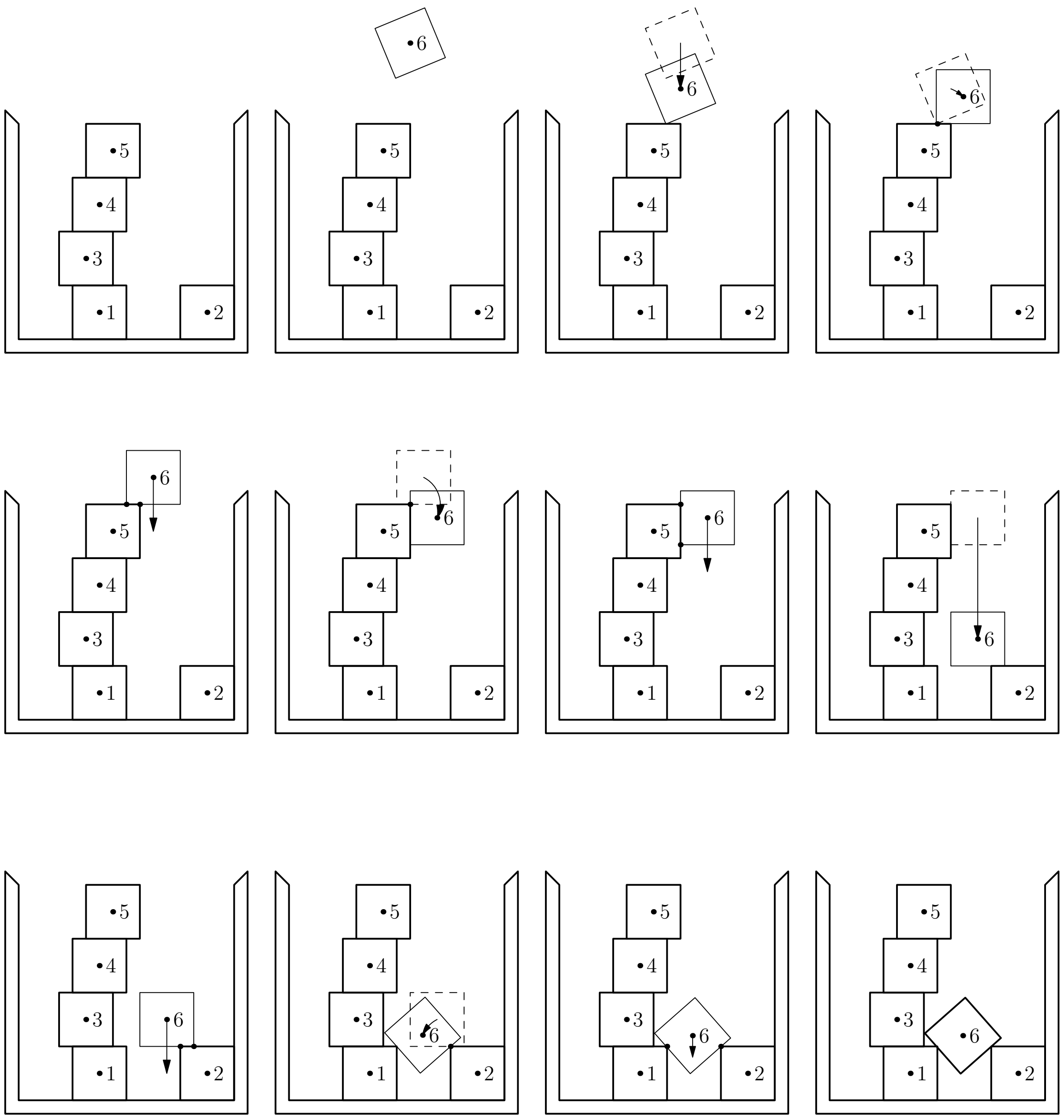

Figure 1: Sketch of the bottom-to-top reconstruction algorithm.

2.2 Implementation

2.2.1 Homemade 2D Implementation

Our first approach was to implement the bottom-to-top reconstruction as discussed above. The overall approach takes the form:

1. Spawn a box with a random rotation
2. Let the box fall until it makes contact with another box or the walls of the container

3. The point of contact will be the pivot point and if the center of mass of the box is to the left of the pivot point, the box will pivot to the left around the point, likewise if the center of mass is to the right of the pivot point.

4. There are two possibilities:

   - The corners of the box hits the edges of the other boxes or the walls of the container
   - The corners of the previously placed boxes hits the edges of the pivoting box

5. To simulate this behaviour, we first take all the corners of the pivoting box and spin them around the pivot point to figure out their closest point of contact (measured in degrees). Next we take all the corners of previous boxes and spin them around the pivot point to also find their closest point of contact to the box. Whichever distance is the shortest will be the first point of contact when the box is pivoting. This only applies if the distance from the corner to the point of contact is less than the side length of a box.

6. Now the box will have two points of contacts. If the center of mass is between these points of contact, the box can't pivot and thus is in a stable position. If the center of pass is to the right or left or either point of contact, the box will then pivot again around this point of contact. Repeat from step 4.

This process is shown in **Figure** 2. The actual visualization of some selected keypoints in a simulation is shown in **Figure** 3.

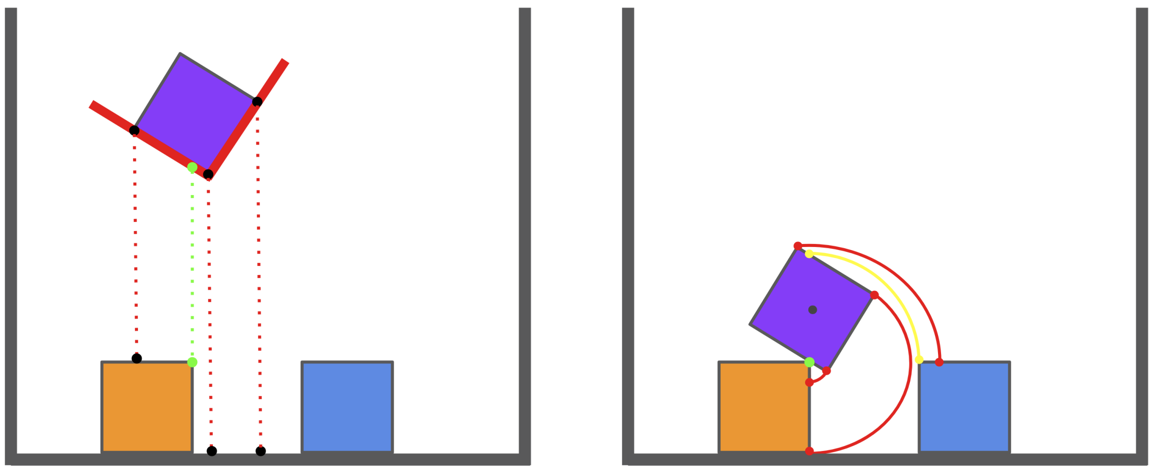

Figure 2: Illustration of finding point of contacts. Left: When a box is spawned, we find two two lines that will collide with other objects and figure out the closest distance to a object from these lines. Right: Illustration of the process with rotating the corners around the pivot point to find the next points of contact.

In this configuration, a box is stable whenever its center of gravity lies between its points of contact. However, this is a very simplified version of the problem and there are many cases where it would be more appropriate to recalculate stability, for instance

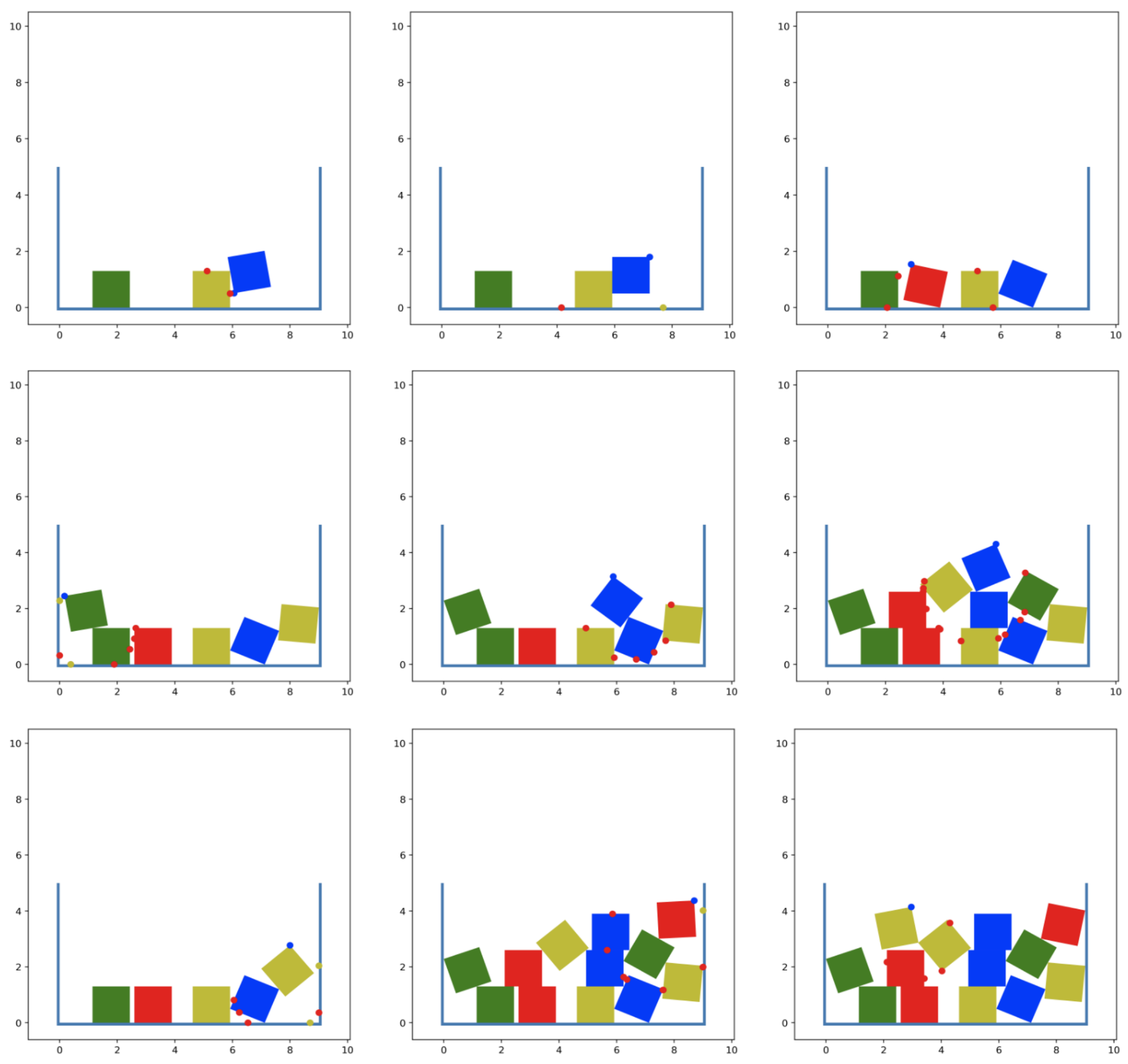

Figure 3: Important keypoints in a random simulation in our homemade implementation of the bottom-to-top approach. Notice how a corner is spun around a selected point of contact to find the location of the box in the next frame

if we have a large tower of blocks that should actually tip over. Also there is no notion of forces in this environment, making the already placed blocks completely rigid objects. To combat this, a proper physics engine, which calculate forces based on mass and momentum must be used to generate more realistic results. An extension of the current code to incorporate forces will be too extensive to handle during the duration of the project. An initial idea is shown in **figure** (4).

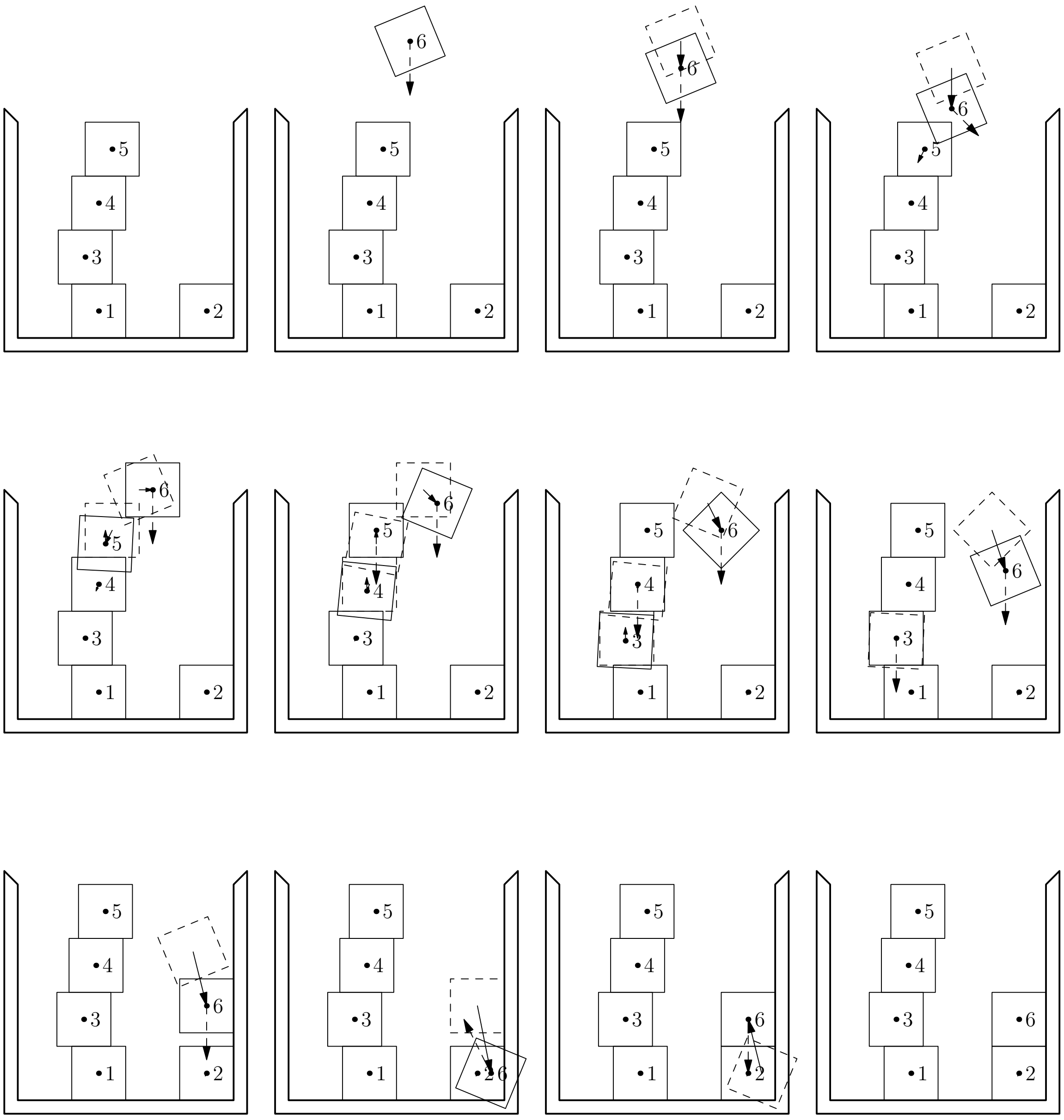
Figure 4: Sketch of the force-based discrete element method.

2.2.2 Implementation using the Box2D Library

To try and incorporate proper physics, writing our own engine didn't seem feasible in a timespan of 5 days so we chose to use the Box2D engine (https://box2d.org).

From the Box2D website: "*Box2D is an open source C++ engine for simulating rigid bodies in 2D*". The engine has python bindings which was perfect for our project.

After getting to know the code, we constructed our environment using this physics engine and were able to do simulations with boxes of varying sizes. Multiple boxes are now spawned above the container and the library will do the physical simulation of them landing into the container and interacting with the other boxes. Once a sufficient amount of boxes are no longer ending up inside of the container, that means that the container is full and we end the simulation, this is verified by visual inspection of the final result which is saved as an image. This is best shown in the youtube video that we have linked in the introduction. **Figure** 5 shows some cases of the simulation running with boxes of different sizes.

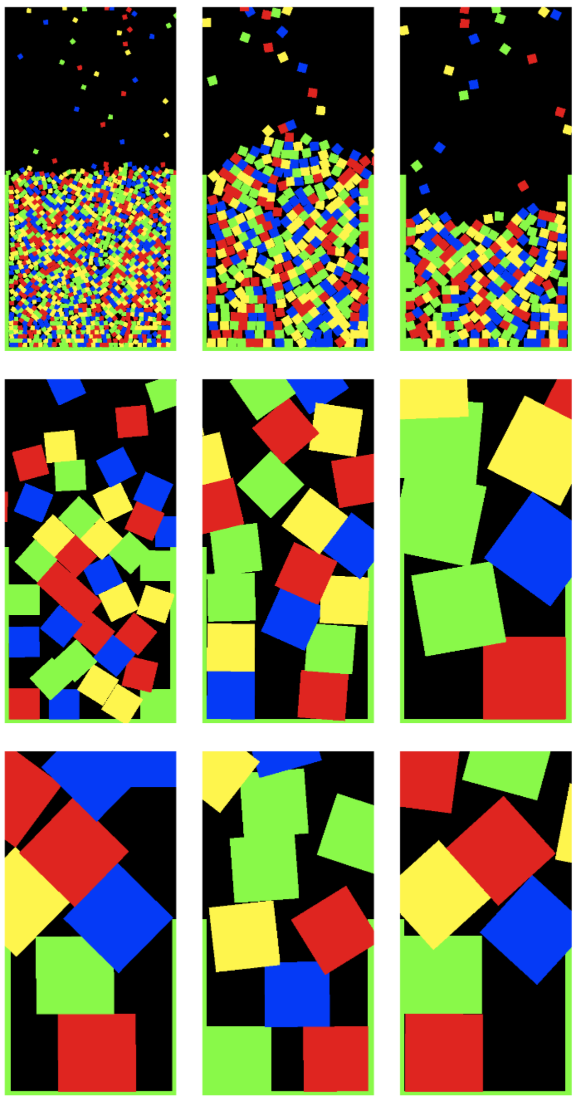

Figure 5: Using the Box2D to simulate throwing in boxes of increasing size. The container is the green open box at the edge of the window

Using this library, we were able to run a lot of simulations in a short time, experimenting with the packing density when we changed the size of the boxes. As seen in the figure above, physics is a bit weird for larger boxes as there sometimes occurs some overlap due to a larger momentum/mass. This however seems to stabilize when new boxes stop being spawned. This library is mainly made for simple games and not running realistic physics simulations. Rigid body physics is a bit more computational as the exact point of contact have to be calculated. For soft bodies, they are allowed to compress a bit and thus the library can be much faster because it can get away with calculating the physics on a frame-to-frame basis. For games, it is more desirable for a physics engine to be fast than accurate, which is the main intent of Box2D.

2.2.3 Tackling 3D using Blender

In order to tackle 3D we could extend our homemade approach to incorporate an additional dimension. Instead of detection collision with lines, it would have to detect collision with planes. Furthermore it will be more difficult to figure out stability and in which direction the box can pivot. In the 2D case it can only rotate clock or counter-clock wise. In 3D it can essentially rotate in every dimension making collisions much harder. The current problems in our homemade approach would most likely also be enhanced by adding more dimensions. Thus an alternative was found in trying to use Blender - a free and open source 3D creation suite. Blender also supports scripting in python so we are able to generate and run experiments from a scripting interface.

Blender is a quite complex piece of software and it takes some time to get started with, which is problematic in a time constrained project. However we were able to model our problem using the software and run a set of simulations with proper physics. **Figure** (6, 7, 8) shows our blender models in action. These are also shown in the youtube video linked in the introduction.

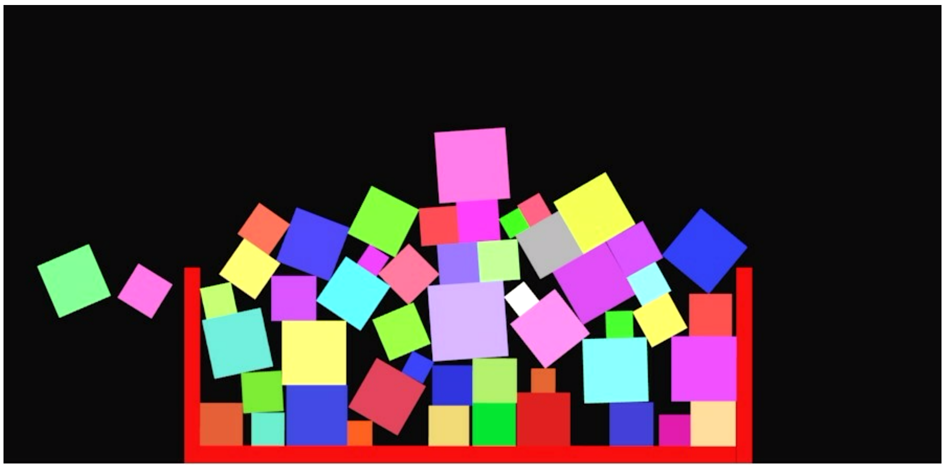

Figure 6: Blender 2D simulation by using very thin boxes between two invisible walls.

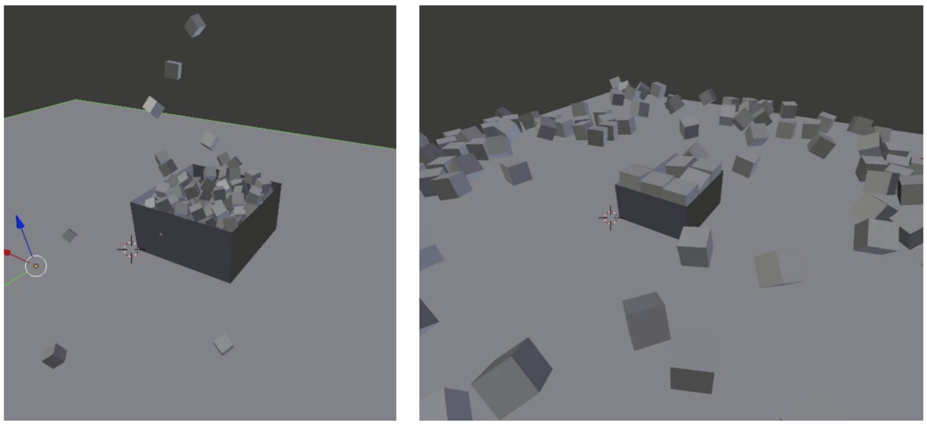

Figure 7: Blender 3D simulation using different box sizes.

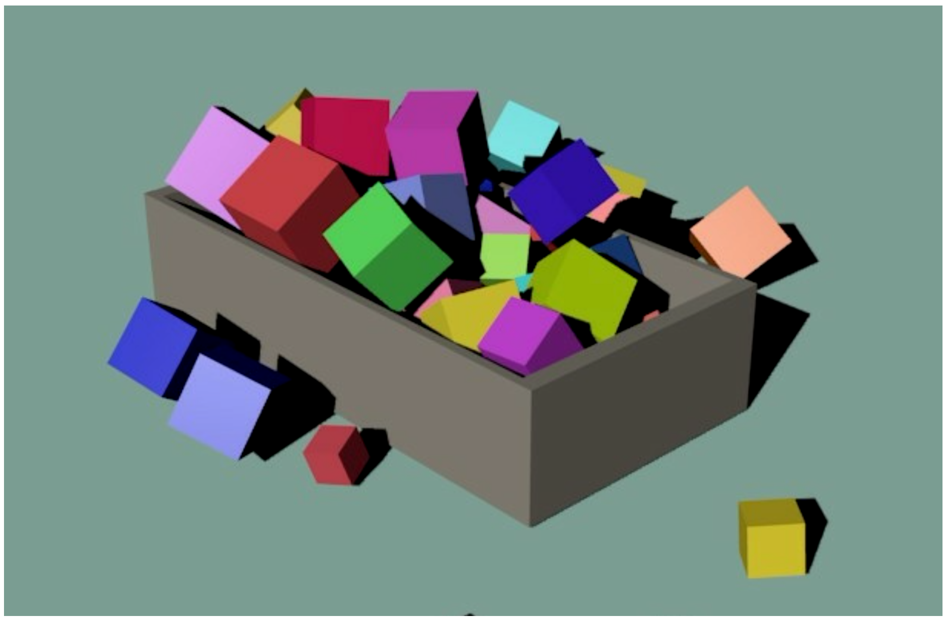

Figure 8: Blender 3D simulation using random box sizes

2.3 Results

Our homemade implementation had too many problems to really give any meaningful result that seemed realistic. Blender did provide a great framework for running simulations in 3D, however it is very computationally expensive to run the simulations and even though we were able to use python scripting to initiate a parametized simulation, we were not able to fetch the box positions at the end of the simulations in time. The box positions are needed to determine if the boxes are inside or outside of the container and thus we were not able to generate an estimate of the packing density for 3D. However initial results from Blender seemed to show, that the packing density of 3D cubes are lower than 2D squares. This seems realistic as an efficient packing requires alignment in more dimensions for 3D than 2D, thus there is a higher probability of getting an inefficient configuration for 3D situations as compared to 2D.

The results for the Box2D implementation is shown in **figure** 9. The results are generated by using a range of box sizes from 1 to 20 and running 30 simulations for each in a 40x40 container. Each vertical group of points corresponds to a set of 30 simulations with a given box size. The main findings are as follows:

1. The average packing density falls as the size of the boxes increase. It makes good sense, that we are in general able to have a more dense packing for small objects when they are randomly thrown in.

2. The variance from simulation to simulations (spread on y-axis), increases as the box size increase.

3. Looking at some of the points we see, that in some cases larger boxes can have a really high packing density. For instance, with a container size of 40x40 and a box size of 20x20, we could optimally fit 4 20x20 boxes inside the container and achieve a packing density of 100%. However throwing in the boxes at random does very rarely generate an optimal configuration.

4. At the end of the graph, we see the points across simulations are forming straight lines as shown in **figure** 10. They essentially just show what happened if we for instance are only able to fit one box inside the container. If there is only a single box inside the container at the end of the simulation, then we naturally get a higher packing density if that one box is larger.

5. Even though the container is 40x40, getting a tight packing of larger objects is very hard when throwing them in at random. Box sizes of 19 and 20 did not have a simulation in which there were more than two boxes fully inside the container.

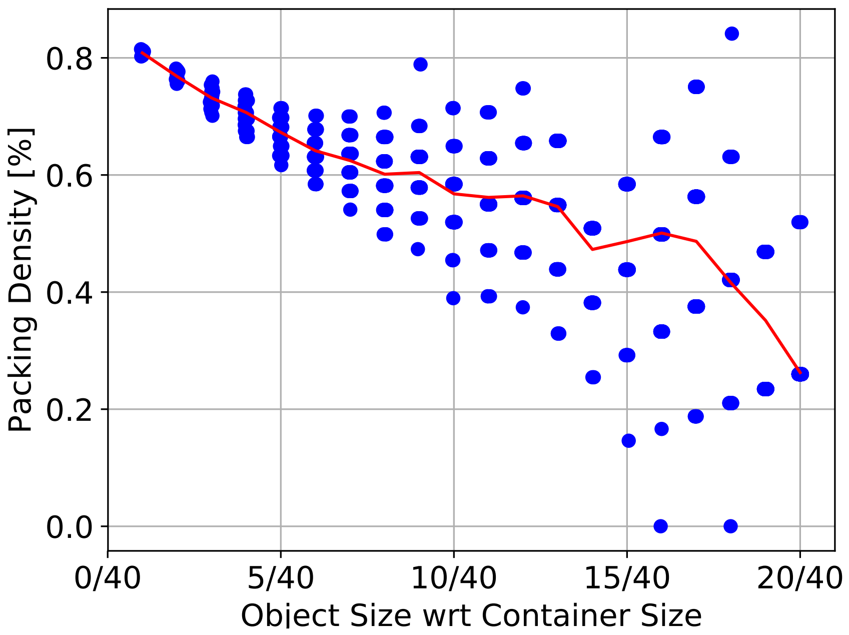

Figure 9: Result of the Box2D implementation

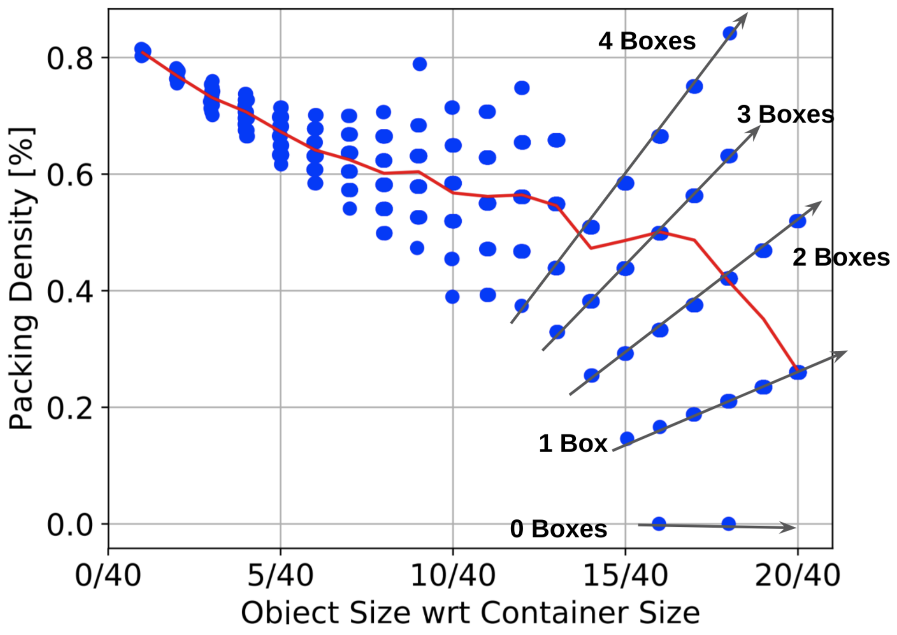

Figure 10: Result of the Box2D implementation with highlighted observation of increasing packing density for equal number of boxes inside the container.

3 Conclusion

In this work we have implemented multiple simulation environments for simulating cubes and squares being thrown into a container at random. We have generated insightful results which proves that our environment behaves as expected. The problem of packing density was tackled in both 2D and 3D and with the environments, we have created a base for future research questions, such as examining alternative shapes and the packing density using a distribution of different sizes of objects.

4 Group work dynamics

To begin with the problem was discussed in the group as a whole. The work was then divided and the group worked individually or in smaller groups of two or three. This made way for different approaches to the problem, theoretical parts as well as modelling work. Work was done in both 2D and 3D, but the latter was excluded as it did not provide any confident results.

5 Instructor's assessment

The group showed a very strong performance during the ECMI Modelling Week in Novi Sad. Their independent and creative way of addressing the problem was quite impressive. Besides analytical approaches the group tried two different numerical simulations, one resorting to using commercial software and another one relying on their own code. Taking all this into account, the group achieved results that were beyond my expectations.

A Order Metrics

In this work or main metric has been packing density, but another interesting metric would be to look at the randomness of the placed objects. The following section is some initial work to answer how neat/aligned the objects are. Unfortunately we weren't able to troubleshoot our implementation in time, so these order metrics didn't give any valuable results.

In physics parlance, quantitative ways to measure randomness in a system are called order metrics. There are various kinds (Torquato 2002). Alas, most assume the system consists of point particles, features bonds (chemical or otherwise) or tends to arrange into a known set of crystal structures. Our system does not have these properties, so we cannot employ them directly.

The best option we could lift from physics is the two-body excess entropy (Truskett, Torquato, and Debenedetti 2000), which has the dimensionless form

$$\frac{S}{k^{\mathrm{B}}} = -\frac{1}{2}\left(\frac{N}{V}\right)\int\limits_{0}^{\infty}\frac{\mathrm{d}}{\mathrm{d}r}\Big(g\log g - (g-1)\Big), \tag{1}$$

where N is the number of particles, V is the total volume of the ambient space, g is the radial distribution function parametrized by the distance r (obtainable through weighted kernel density estimation (Kiiskinen 2018)) and k^{B} is the Boltzmann constant. However, this does not consider the orientation of the bodies. To do that, we could use the orientational correlation function (Donev et al. 2006), but it is not obvious how to incorporate it yet.

While the (unnormalized) radial correlation function says

$$g_2(r) = \langle\delta(r-r')\rangle_{r'}, \tag{2}$$

the orientational correlation function

$$g_m(r) = \langle\cos m(\theta-\theta')\rangle_{\theta'}, \tag{3}$$

where m is the degree of the symmetries as in (Donev et al. 2006). We conjecture that for squares $m = 4$ and for rectangles $m = 2$. More is in (Saintillan and Shelley 2007) and (Stoyan and s 1991). Unfortunately combining these is not discussed anywhere.

References

Donev, Aleksandar et al. (2006). “Tetratic order in the phase behavior of a hard-rectangle system”. In: *Physical Review B* 73.5, p. 054109.

Kiiskinen, Sampsa (2018). “Variations of a Discrete Element Method for Friction between Shearing Brittle Surfaces”. MA thesis. University of Jyväskylä.

Pöschel, Thorsten and Thomas Schwager (2005). *Computational Granular Dynamics: Models and Algorithms*. Scientific Computation. Springer.

Saintillan, David and Michael J. Shelley (2007). “Orientational Order and Instabilities in Suspensions of Self-Locomoting Rods”. In: *Physical Review Letters* 99.5, p. 058102.

Stoyan, Dietrich and Victor Bene ˇ s (1991). “Anisotropy Analysis for Particle Systems”. In: *Journal of Microscopy* 164.2, pp. 159–168.

Torquato, Salvatore (2002). *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*. Springer.

Truskett, T. M., S. Torquato, and P. G. Debenedetti (2000). “Towards a Quantification of Disorder in Materials: Distinguishing Equilibrium and Glassy Sphere Packings”. In: *Physical Review E* 62.1, pp. 993–1001.

Visscher, William M. and M. Bolsterli (1972). “Random Packing of Equal and Unequal Spheres in Two and Three Dimensions”. In: *Nature* 239.5374, pp. 504–507.