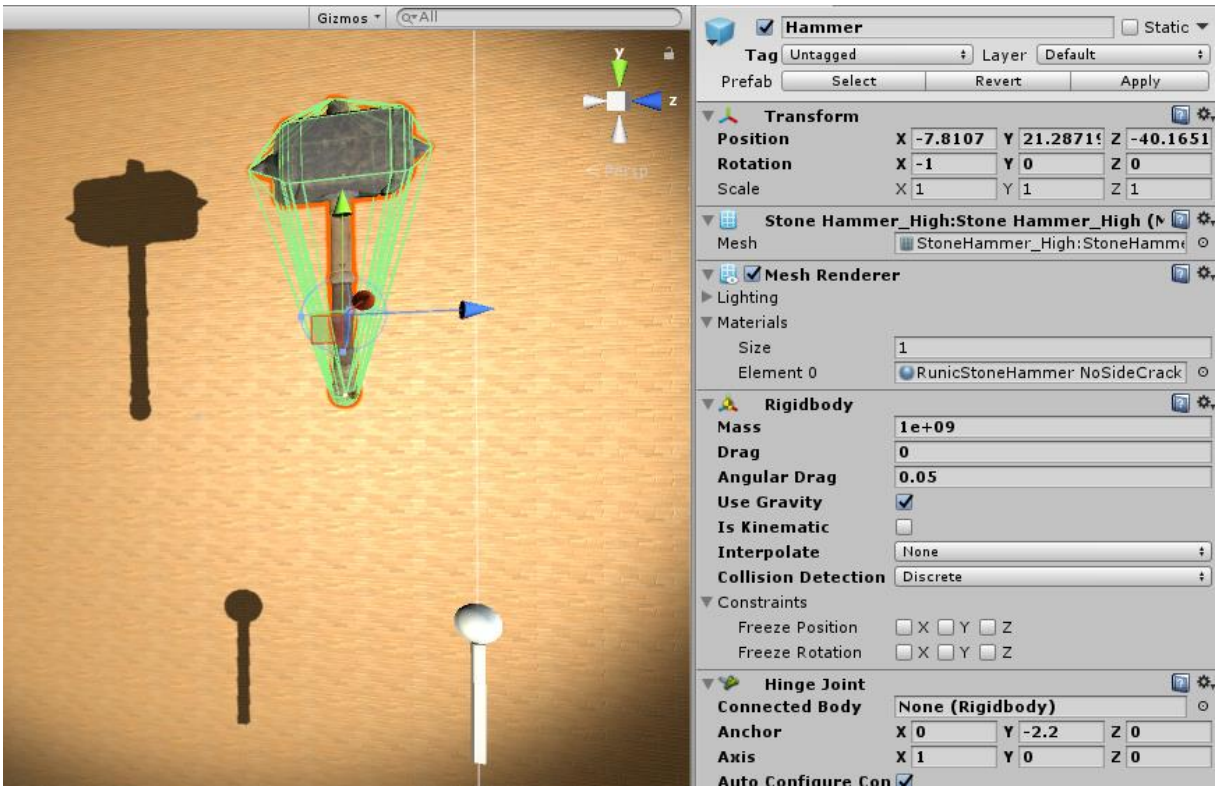
Report

Introduction

Before implementing the machine, I firstly drew out the steps on a sheet of paper. After I eliminated ideas which would make for a yearly project, I started the implementation, trying to keep as close to my design as possible.

1. The Hammer Bash

I wanted to have the start different from any other subsequence in the project, to ensure that it is clear what starts the whole chain of events, deciding in the end on a hammer bash. The hammer, basing on a hinge joint, falls down under an angle (starting with minus 1 degree on the X axis) and collides with the ball, sending it flying. What is interesting, is that the distance covered by the ball would not increase after the difference in mass went above 1 thousand. Still, I left it high to ensure that the result will be the same most of the time. The ball is not frozen in the X axis, making this a purely physical event.



1 The values of the hammer.

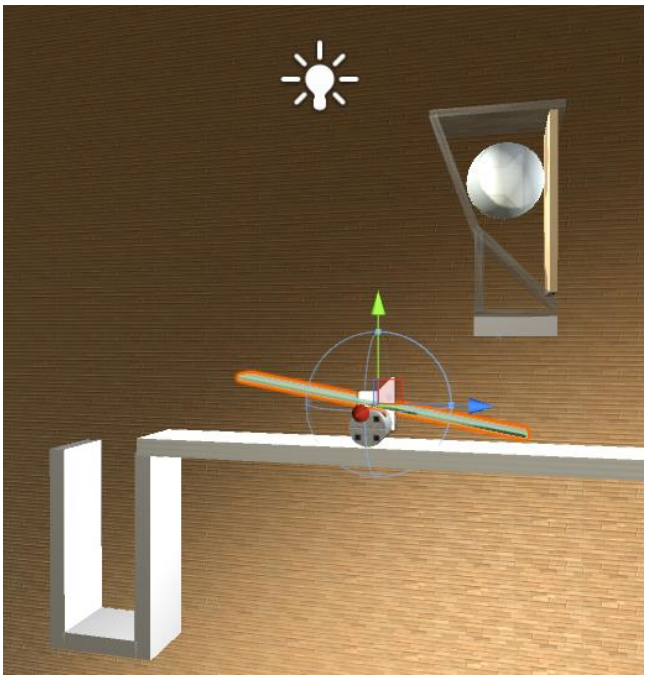
2. Release the ball

The ball from the previous event falls to a platform with a seesaw on it. When it lands, it counterweights the seesaw so the second end of it hits a switch (colouring it red), releasing a ball. The previous ball is then collected into a “sphere collector”, while the second ball continues.

The hatch behind of which is the second ball operates on a hinge joint. The switch simply changes the “isKinematic” value of the hatch, to make it fall down.

What I did previously is that I focused on making the second ball release itself by changing the kinematic setting of it, so that it would make the hatch fall down just due the use of gravity. Unfortunately, in the end the hatch would usually fall by itself and defeat the purpose of this. Therefore I focused on affecting the hatch with the switch.

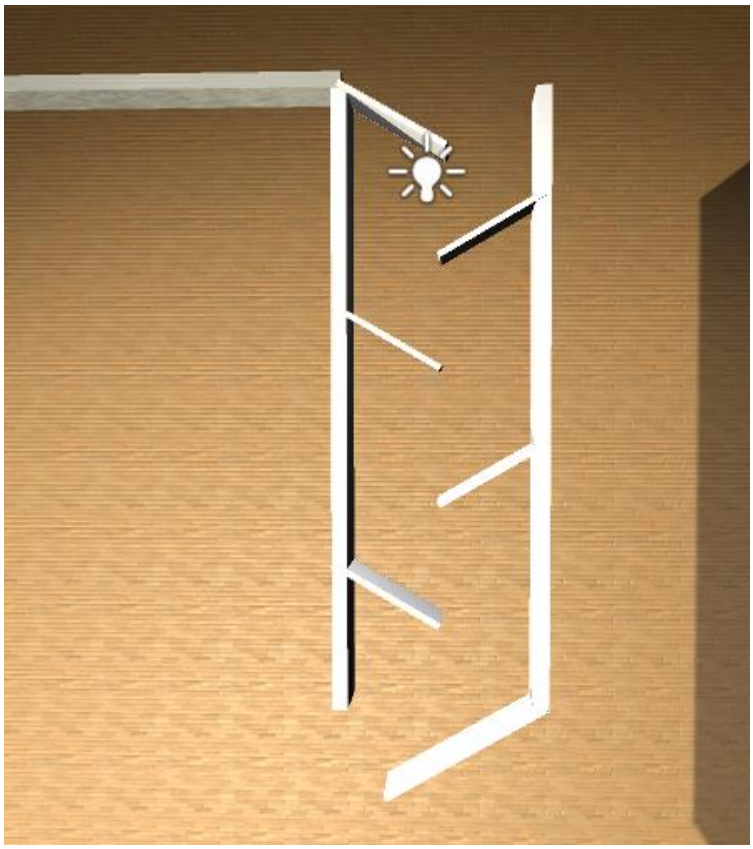
I also had to freeze the X axis of the ramp, as sometimes after hitting the switch, it would fall down from the platform.



2 The second step.

3. Fall down to pings

The released ball falls through what I called “fallers” to the next step. These are just angled platforms that light up when the ball collides with them. Except of being a nice touch, I added them to have control of how the ball will exit the fall.

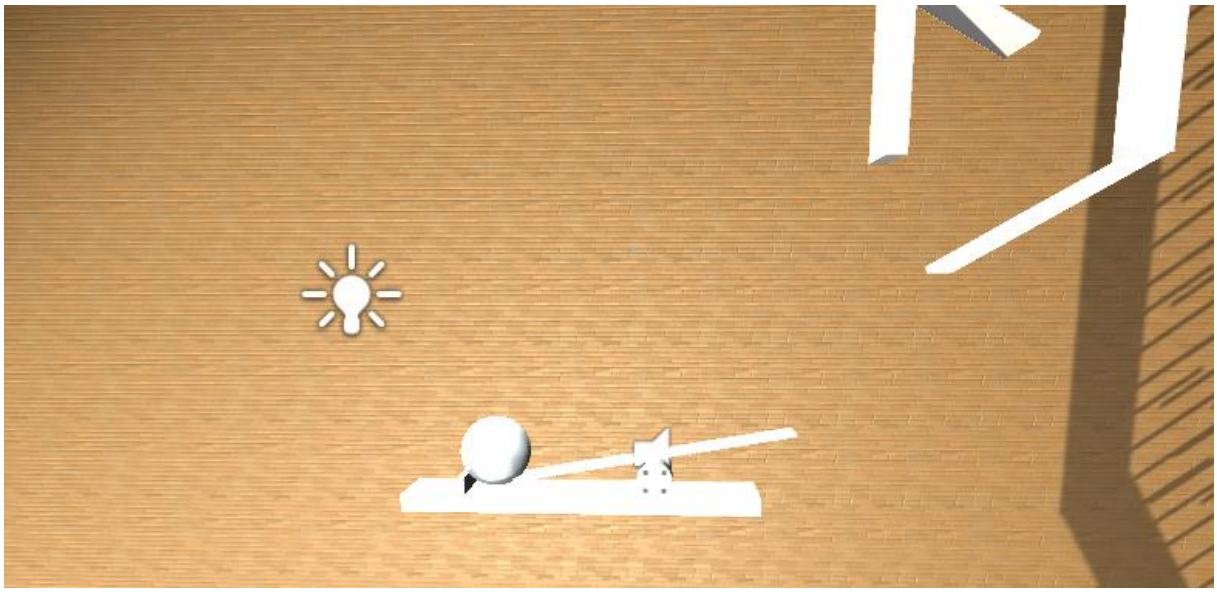


3 The third step.

4. **Release another ball**

The ball that falls from the previous step counterweights another seesaw and releases yet another ball by moving it upwards with the seesaw. This time there are no switches involved, and the ball is only held by a small wall in front of it. I made the wall small enough to release the ball, but also high enough to hold the fallen ball.

The seesaw is held in the X axis as it used to drop sometimes as well.

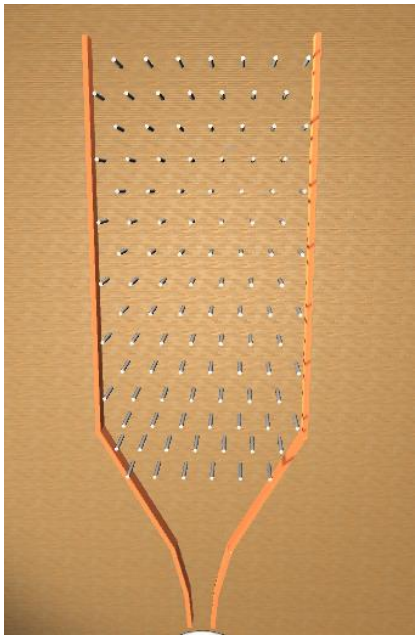


4 Ramp releaser.

5. Fall down to bowls

The released ball falls down through a serie of “pings” (I have no clue how is it called normally), showcasing the fall and collision physics before entering the next step.

In theory, the collisions with pings seem to be random, but as the ball always begins the fall from the same position (ramp above), the fall pattern stays the same. If I needed to make it more randomised, the begin point should be more varied.

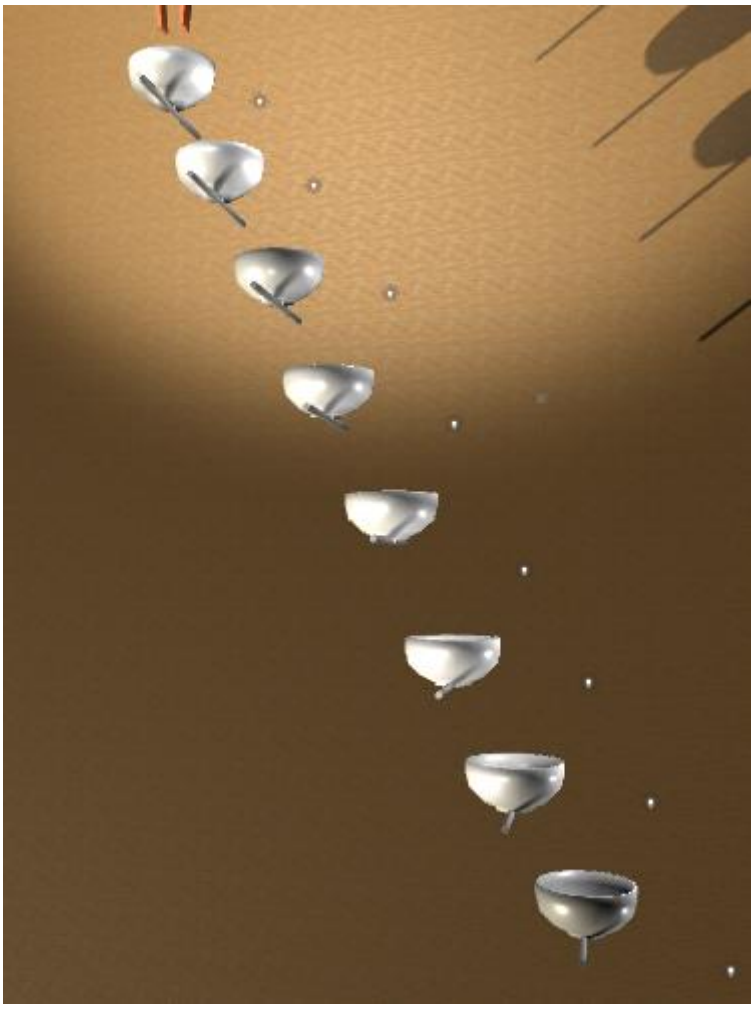


5 Pings.

6. Bowls!

The ball falls through a serie of bowls, which rotate on collision.

Sadly, I initially wanted to use hinge joints to have the bowls rotate due to physics (when the ball falls down and counterweights the bowl). Unfortunately, when I set a mesh collider with a hinge joint, it would stop detecting collision. I had to therefore set up a script rotation.

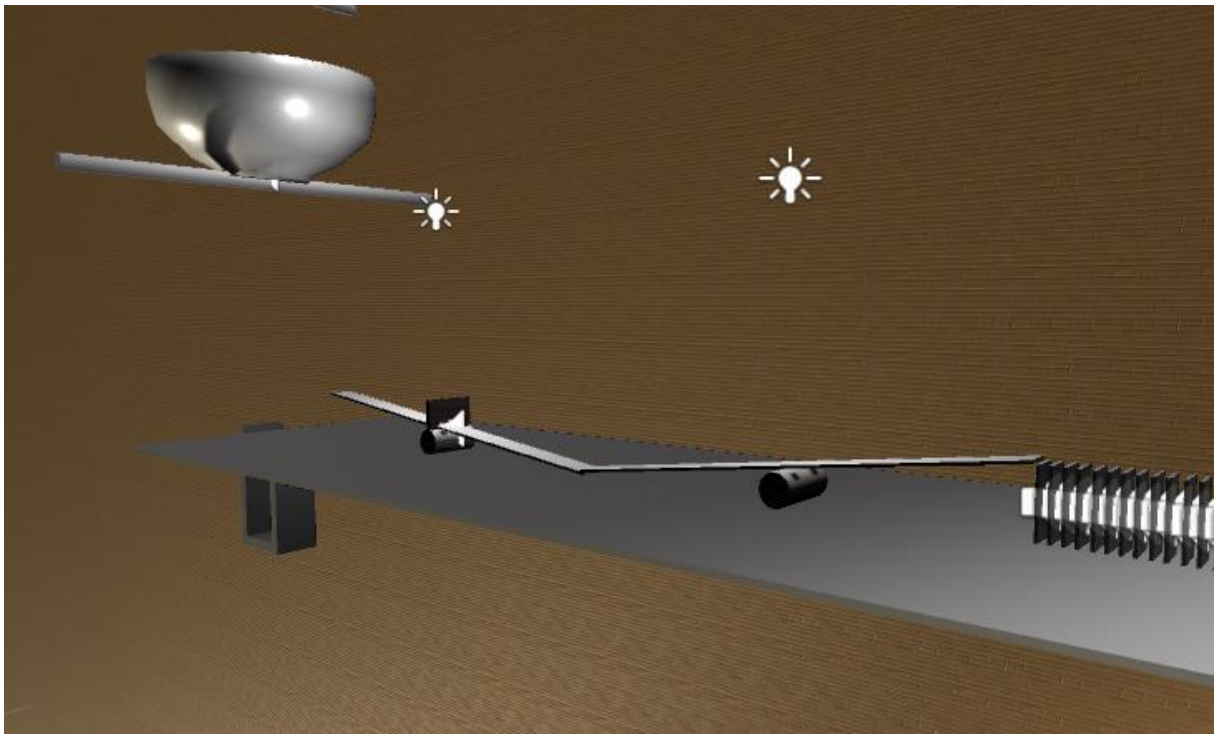


6 Bowls.

7. Domino

The fallen ball affects two seesaws, the last one pushing the first domino, causing a chain of falling dominos.

I previously made the dominos to have around 3000 of cubes. The amount of calculations that resulted made me change my decision to around 370, making the distance between the dominos a bit bigger.

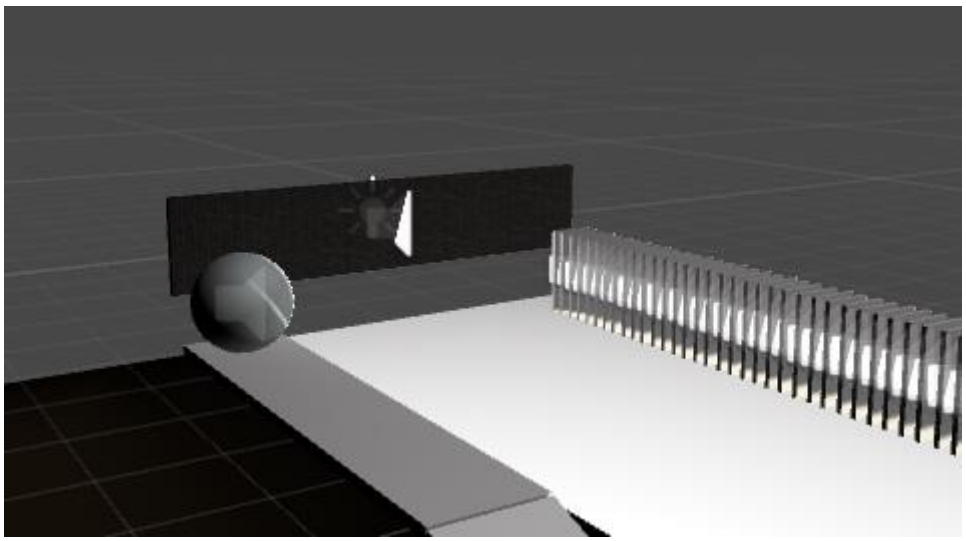


7 The start of the domino.

8. Push the ball

The last fallen domino pushes a platform, which turns and pushes a ball.

The platform had to be not too wide to ensure that it does not hit the domino when it turns around. Surprisingly, the last domino, the platform, and the ball each have a mass of 1, as I did not have to set the last domino's mass value to anything strong.

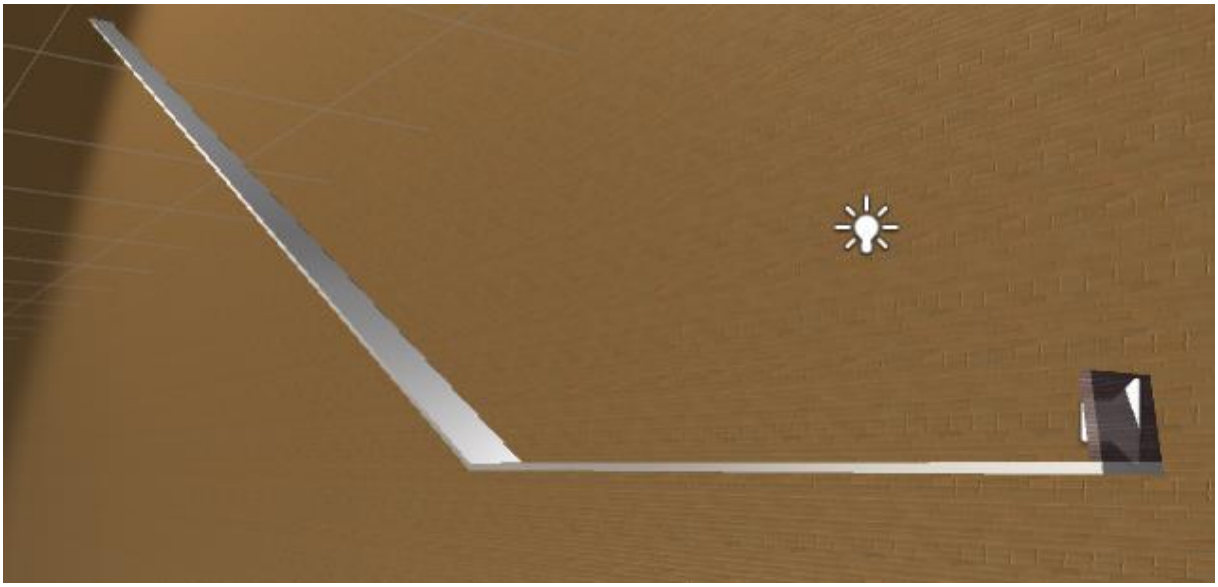


8 The platform.

9. Fall down to big balls

The ball slides down and hits a switch on a wall, setting the first wrecking ball's kinematic setting to false.

This step was added to let me move the ball around a bit, as I wanted to see how well friction works in Unity. As I found that the default terminal velocity is low, I made the angles a bit low as well as there was no point in setting them to 80 degrees for example.

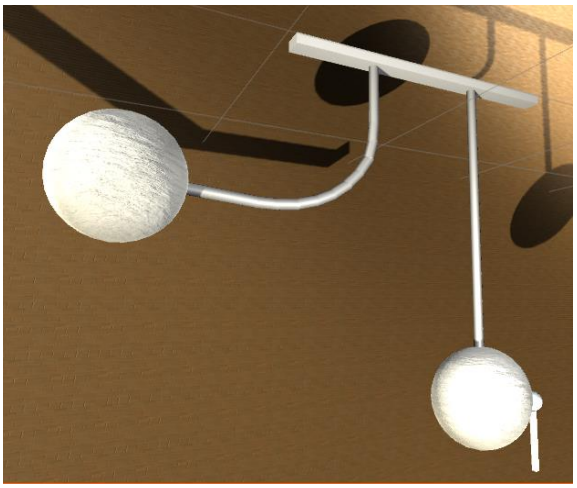


9 The slide with the switch.

10. Hit the big balls

The angled wrecking ball falls down, hitting an another wrecking ball.

The wrecking balls' ropes are a couple of cylinders connected with hinge joints, as I found there is no easy way of creating a rope in Unity. Interestingly enough, the mass of the balls is set to 1, even though they appear as gigantic in the project.



10 The wrecking balls.

11. Push a ball to a cannon

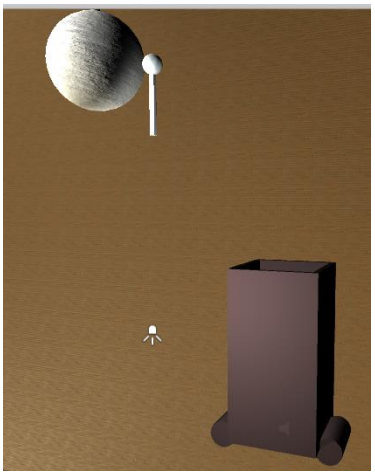
The second wrecking ball pushes a ball from a stand to a cannon.

It took me quite some time to get the ball to fall properly into the cannon, as the previous collision has various end results. In the end, I made the cannon 3 times as large as it was in the beginning to make sure that the ball lands in the cannon.

12. Shoot the cannon

The cannon detects the collision with the ball and shoots it.

The ball that falls into the cannon has force applied to its Y axis when it collides with the bottom of the cannon. Since it falls into the cannon, I had to lock the rotation of the ball to make sure the cannon does not end up firing 10-15 times before the ball leaves the cannon.

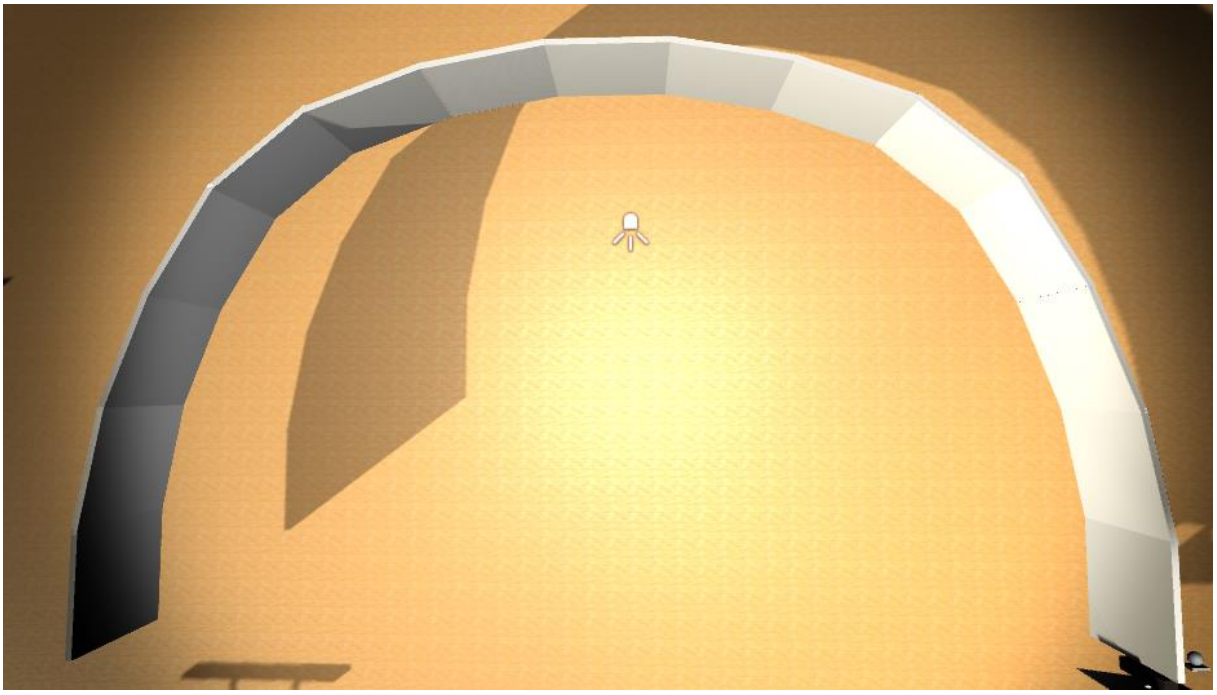


11 The cannon and the ball stand.

13. Slide the shot ball through an arc

The shot ball slides through an arc, making use of the friction system.

The parts of the arc change colour when they collide with the ball, to see the flow of the ball. That is why one of the parts does not change its colour to blue – it is not animated.

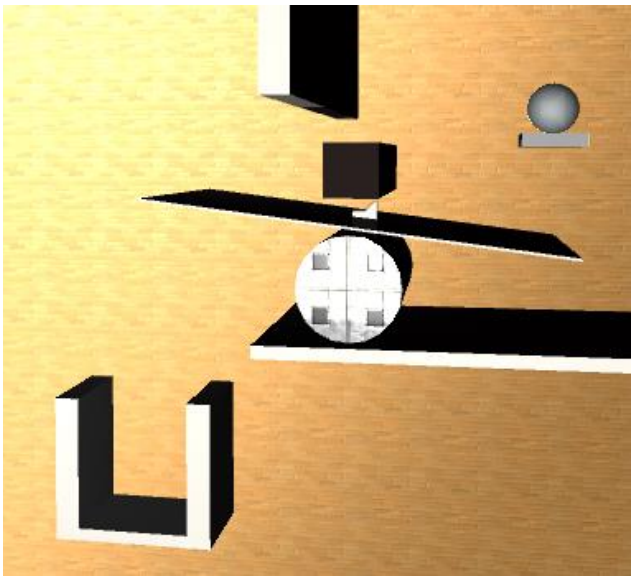


12 The arc.

14. Release another ball

The shot ball hits a seesaw, releasing a ball from a wobbly stand.

I wanted to test the Unity's physics system and see if a ball can stand on the middle of a very small stand. It turned out that it can, though it costed me a lot of time to set the ball perfectly in the middle, as even the slightest floating-point change will cause the ball to fall one way or another.

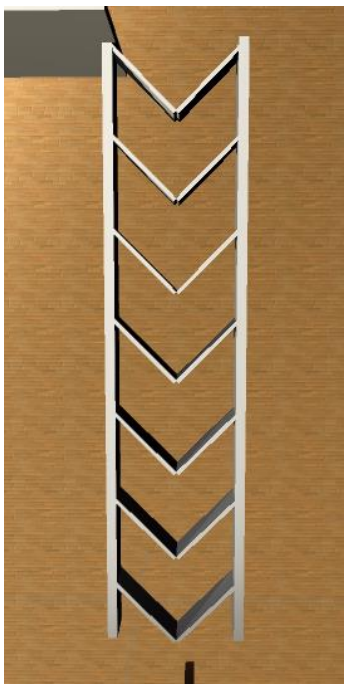


13 Stand.

15. Fall through sliders

The released ball falls through what I called “sliders” to the next step.

The sliders are hinge joints which have their kinematic setting turned to false when a ball collides with them, creating a nice imitation of hatches. I initially wanted to make them return to their initial position as if they were springs, but after battling for a long time to have them not escape from the wall I returned to using hatches.

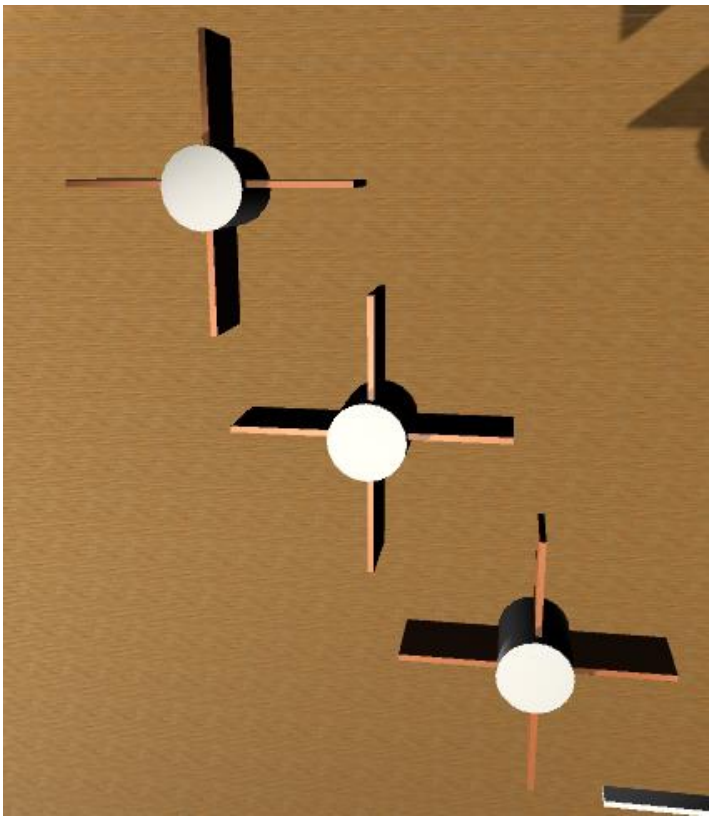


14 The sliders.

16. Clockwork machine

After falling from the “sliders”, the released ball goes through a clockwork machine.

The clockwork machine is the main factor of how the rest of this project works, as it rotates independently from the rest. Therefore, sometimes the ball will go through it very quickly, making the animated camera lag behind, and sometimes it will have a bit of trouble to finally land on the platform, making the camera go too fast. This caused me a lot of trouble when I was animating the camera, and the current animation should be able to catch both exceptions. I also made the rotation a bit slower from the initial design.



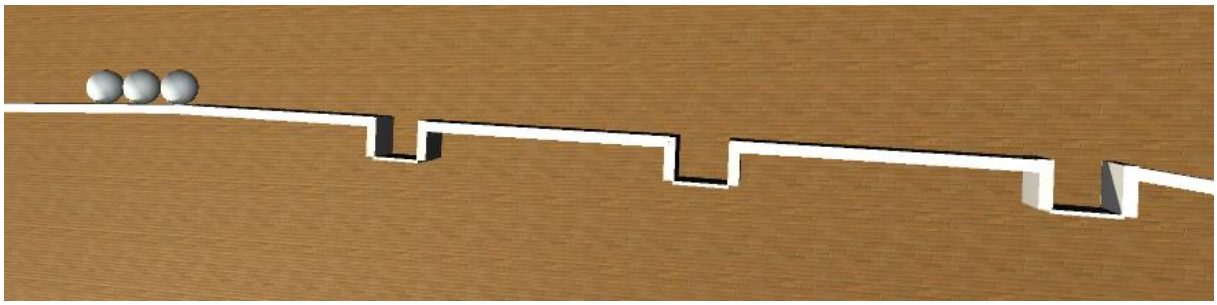
15 The clockwork machine.

17. Slide with balls

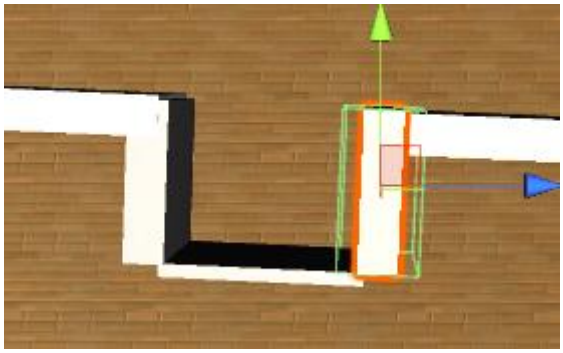
The ball hits 3 balls which are smaller in mass, which later on drop into holes in the platform and create a path for the ball fallen from the clockwork machine.

This step is also quite a trouble not only for the camera, but also for the project, as the clockwork machine drops the ball with various speed – sometimes the fallen ball will hit the others with a lot of force, and sometimes with a tiny amount of velocity.

To ensure that the balls actually fall, I set the mass of the three balls to 0.1 (10 times lower than the falling ball), and also made the box colliders for the gaps in the path to be a bit larger than their meshes, as sometimes the balls would get stuck on one of the balls in the path.



16 The path.

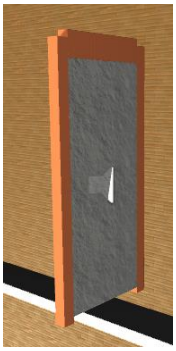


17 The colliders.

18. Go through a hatch

The ball goes through a hatch to the next step.

The hatch uses a hinge joint, and it also has smaller mass than the ball, to ensure that it can be pushed by the ball, as that was an often problem with my first implementations of it.



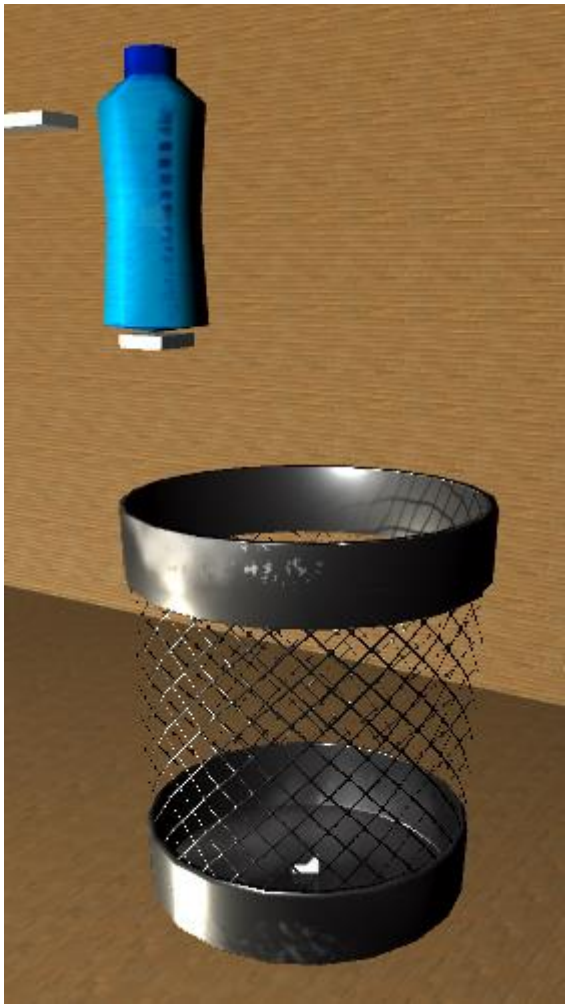
18 The hatch.

19. Push a bottle into a garbage bin

Finally, the ball falls and hits a bottle, putting it into a garbage bin, as well as the ball (in most cases).

To ensure that it falls properly, the bottle has a mass quite lower from the ball's.

When the water bottle hits the container (by checking mesh collisions), a win sound is played with a restart menu later on.



19 The end.

Summary

In the end, I managed to finish my goal of having most of the events being caused by collision. Unity's physics system is quite intuitive and most of the times behaves quite like the real life. With a big enough team and enough time, I believe that many of the current scripts used for triggering a movement by collision in Unity games could be just replaced by adjusting values for a proper physical collision.

I used some of the code from Unity's documentation, either by direct usage (like the audio) or by having my own implementation of the code, as I could not find better ways of implementing needed functions.

References

- **Assets**

Bowl: Kitchen Props Free (2017). JAKE SULLIVAN. [online] Available at: https://assetstore.unity.com/packages/3d/props/interior/kitchen-props-free-80208 [Accessed 20 May 2018].

Hammer: Runic Stone Hammer (2016). FERNANDO FAGUNDES. [online] Available at: https://assetstore.unity.com/packages/3d/runic-stone-hammer-67297 [Accessed 20 May 2018].

Materials: Prototype Materials Pack (2016). BURAK TABAN. [online] Available at: https://assetstore.unity.com/packages/2d/textures-materials/prototype-materials-pack-65136 [Accessed 20 May 2018].

Trash bin: Trash Bin (2017). YGS ASSETS. [online] Available at: https://assetstore.unity.com/packages/3d/props/furniture/trash-bin-96670 [Accessed 20 May 2018].

Water bottle: Small Survival Pack (2014). JOSCH. [online] Available at: https://assetstore.unity.com/packages/3d/props/small-survival-pack-20565 [Accessed 20 May 2018].

- **Sounds**

Background music: Funny Bone (2017). MITCH MUSIC. [online] Available at: https://soundcloud.com/mitchmusicco/funny-bone [Accessed 20 May 2018].

Bottle sound: Plastic Bottle Hit (2013). WarcakeStudios. [online] Available at: https://freesound.org/people/WarcakeStudios/sounds/173598/ [Accessed 20 May 2018].

Cannon sound: Cannon1.wav (2013). Isaac200000. [online] Available at: https://freesound.org/people/Isaac200000/sounds/184650/ [Accessed 20 May 2018].

Creak sound: cupboard creak 3.flac (2006). tim.kahn. [online] Available at: https://freesound.org/people/tim.kahn/sounds/24244/ [Accessed 20 May 2018].

Domino sound: Dropping tiny pebble in still water surface (2012). jorickhoofd. [online] Available at: https://freesound.org/people/jorickhoofd/sounds/160119/ [Accessed 20 May 2018].

Drop sound: LightBulletPing.mp3 (2016). wilhellboy. [online] Available at: https://freesound.org/people/wilhellboy/sounds/351369/ [Accessed 20 May 2018].

Hammer bash sound: Ambassador-6.wav – Saturated (2014). Seidhepriest. [online] Available at: https://freesound.org/people/Seidhepriest/sounds/232014/ [Accessed 20 May 2018].

Hammer swing sound: Bamboo Swing, A4.wav (2017). InspectorJ. [online] Available from: https://freesound.org/people/InspectorJ/sounds/394421/ [Accessed 20 May 2018].

Metal turn sound: 95SwingDrum.wav (2016). fschaeffer. [online] Available from: https://freesound.org/people/fschaeffer/sounds/337912/ [Accessed 20 May 2018].

Pinball sound: Classic Pinball Gameplay (2017). theshaggyfreak. [online] Available at: https://freesound.org/people/theshaggyfreak/sounds/404144/ [Accessed 20 May 2018].

Switch sound: Button 05.wav (2014). JarredGibb. [online] Available at: https://freesound.org/people/JarredGibb/sounds/219476/ [Accessed 20 May 2018].

Win music: Jingle_Win_01.wav (2015). LittleRobotSoundFactory. [online] Available at: https://freesound.org/people/LittleRobotSoundFactory/sounds/270545/ [Accessed 20 May 2018].

Wrecking ball collision sound: fireworkblast.wav (2012). nfrae. [online] Available at: https://freesound.org/people/nfrae/sounds/167242/ [Accessed 20 May 2018].

- **Tutorials**

Unity3D, 2018. *AudioSource.Play* [online]. Docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/AudioSource.Play.html [Accessed 20 May 2018].

Unity3D, 2018. *Canvas* [online]. Docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/UICanvas.html [Accessed 20 May 2018].

Unity3D, 2018. *MonoBehaviour.OnCollisionEnter(Collision)* [online]. Docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnCollisionEnter.html [Accessed 20 May 2018].

Unity3D, 2018. *Rigidbody.AddForceAtPosition* [online]. Docs.unity3d.com. Available from: https://docs.unity3d.com/ScriptReference/Rigidbody.AddForceAtPosition.html [Accessed 20 May 2018].

Unity3D, 2018. *Transform.Rotate* [online]. Docs.unity3d.com. Available from: https://docs.unity3d.com/ScriptReference/Transform.Rotate.html [Accessed 20 May 2018].