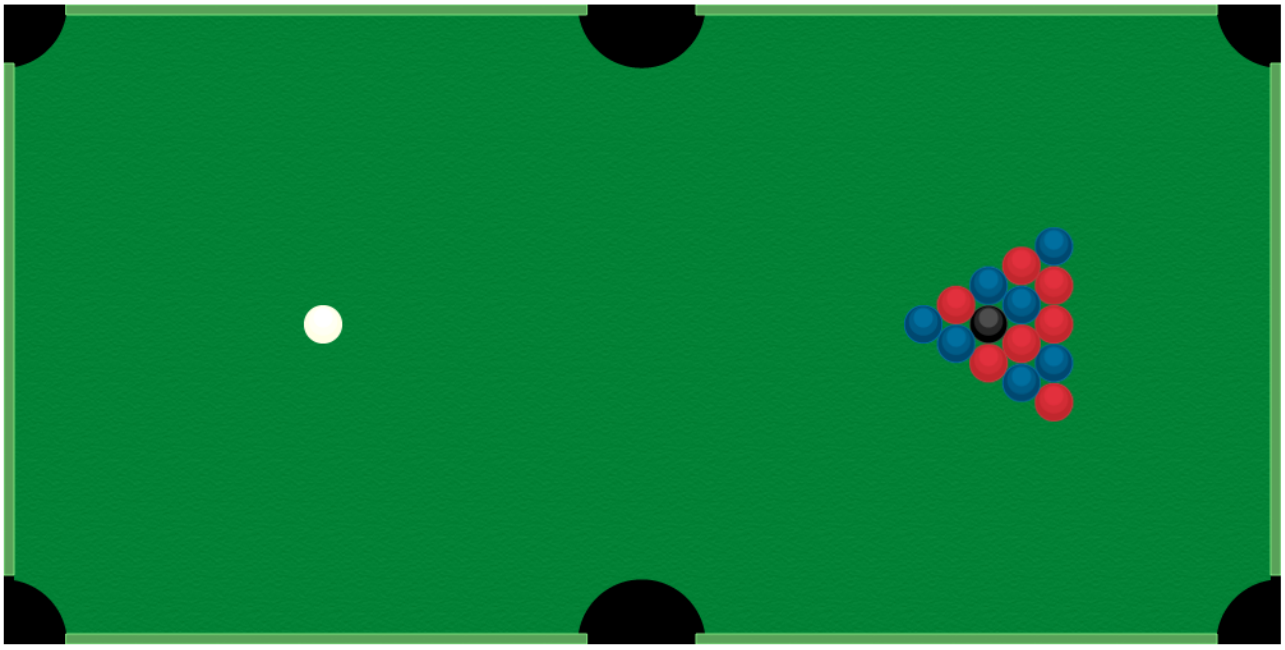
Codeari - Bar pool

Group 8

Janne Tallqvist, 0372294

Sami Tarkiainen, 0358658



Introduction

The game idea came pretty fast. First we discussed if we would implement a tower defense but then we found the idea about the pool game from the internet. After that we started to think more about the idea and what interesting components we would add. There are lots of pool games that are quite normal and all have same idea. Janne had an idea that the pool game is in the bar and both players drink booze at the same time they play. So the game would become harder and harder after every hit. Obviously we would just simulate the drinking process and make hitting the white ball harder etc. But of course you can enjoy beverages of your choice while playing.

We looked very forward to this code camp because we had about two years experience developing with javascript. So we hoped it would give us an advantage in this code camp.

Game mechanics

Code camp's base idea was to build a game with javascript and of course we started to do so but in the beginning of the idea we realized that we would need something more than just our coding to achieve the goal. Of course first we started to search frameworks and physics engines what we would use. We found many engines but after 20 minutes or less we bypassed it engine by engine. In some point we thought building our own physics engine, but we came up to that it would take too long. Fortunately, or maybe not, we found Box2d physics engine which was now on javascript, called box2d web.

With javascript we used obviously HTML5 and canvas. That was so straight forwarded and tiny bit of the code we don't have anything to mention about it. So almost everything interesting was in the box2d web and we were counting on it.

The game

First we tried to understand something about the box2d and we practically just played with it. We downloaded a demo and watched a tutorial. The demo for box2d had some boxes and balls what you could move around with your mouse. So basically it demonstrated the capabilities of the engine.

At this point we had some sort of idea what we would do and how it could be achieved. The coding began by simply reverse engineering the existing demo of box2d engine. We picked the elements and operations that we would need in our game. To set short the most important things in the physics engine for us was the functions for elements colliding and bouncing from each other. Gravity and so for ever was not included or needed, instead we needed to take count "air resistance" what would simulate the resistance between the balls and the table surface.

We have something!

But. We managed to build the game table and set couple of balls to the table. We were not happy with the fact that when you clicked a ball with mouse the ball would follow the mouse. This was one of our biggest problems. But at this point we didn't focus on that problem. We just wanted to get the physics close to realistic. Balls were too fast and they didn't bounce enough from each other.

And the shit hit the fan with a big one. The box2d engine is based to metric system. What was just a huge pain in the ass. We are familiar with using pixels and imaginary values, but on this case all this digital value mess would be in metric scale. Which in other hand could be very handy, if it just could be described somewhere so that an human could understand it. But furthermore we set the scale so that 400 pixels would represent one meter.

From here we can get nicely to another problem lead by the metric/pixels scaling thing. The size of the table. In real life the table is somewhere around 100 x 50 inch, so 2,5 x 1.25 meters. So in theory the table width would use this formula to get the width on pixels: scale*2.5 = 1000 px. Yes makes sense. But when you place the element on the table and you want it to be in the center so you would guess that it's just using scale*1.25 but no it is just something else. We just used try and error to get elements on the right places.

One thing that the metric scale gives to the engine is the fact that you can use real weights and acceleration values. What we didn't fully understood either. We have a guess that one ball weighs around 400-500 grams, but who knows what it is. Weight correlates to the deceleration on bounciness of the balls. Even the balls are too heavy, but they seem to work quiet nice and it is close enough.

Images to nice looking balls

In basic mode the box2d uses development screen to show the structure of the elements or in this case bodies. But of course they look dull and all the balls are in same colour. So they need some imaging to cover the body. And yes you guessed right this was not so easy task.

We googled furiously for some sort of help to this problem. In one example person on the internet had set the information of the image to the userdata of each body. This is a place where user/developer can store some additional info related to the body. Finally in the stage were the engine updates the body information and draws it on the screen. it reads the userdata and the position of the body and draws that image in the same coordinates. So now we had then nice looking balls!

At this stage it had come clearer that the box2d engine is extremely good but bad. Good was the fact that it was working and bad was lack of documentation, no API was found.

Moving the balls

In this stage we had managed to get something working. We had table, balls, images and physics were relatively in good shape. At this point clock was around 6:00 pm.

Our firs idea was to make the ball hit mechanism to work same way around as in Angry Birds. So that when you click and hold on a ball and drag away from it and on release the ball would move to the opposite direction of the drag. The length of the drag would simulate the force of the hit. We started with a ball following the mouse cursor.

In solving this problem the internet proved itself to be completely useless. It was like we was first humans ever trying to implement this kind of a feature in a javascript and HTML5 based game, using box2d engine. We tried several different approaches to this problem. All of them was completely useless. At around 9:00 pm, 3 hours later we had nothing. We were thinking to give up. But somehow continued the battle against the mental windmills. Shortly after this mental meltdown we figured out that we need to create this thing.

So we started working with it. Method was pretty much try and error. First we wanted to set the ball so that when you click and hold to it and then drag the ball would not move. This was a relatively easy task. next thing was to figure out what would push the ball forward, or any direction at this point. Normally the ball would follow the cursor and using those function would not give any good results. Now when there was no API documentation we were again mentally killing ourselves. But fast and furious googling gave the result: impact. How to use this was a task to find out. Finally we figured how to use the impact function.

Now again we were at a point where all methods were clear and the only thing to do was to write it. So we wrote. And some errors were made but we resulted with a ball that you can click, hold and drag. When released it went always to the same direction with same speed. It worked, sort of. Next was to track the point of the original click and then compare it to the point where it was released. So that we could then calculate from these points the direction and force vector needed to propel the ball. This was a problem too. Where to store the information now so that it would be usable several frames forward. And yes every frame change zeros everything. We did something that is not so valid but worked in this case. Public variables. So in this face we have managed to track mouse click, drag and release coordinates. We used some mathematics to figure out the vectors from the coordinates. No big deal with that. We made some configurations to the impact force and maximum force that can be applied and yes the ball moved how we wanted it to move. At this point clock was round 12.00 pm. So it had taken roughly six hours to get the drag and hold to work like we wanted. And yes we made it for ourselves.

Nice to know: We were using Sami's Sony Vaio which has a touchscreen. Janne was using both Sami's computer and his own MacBookPro without touchscreen. At this point Janne tried to use his Mac as a touchscreen device, multiple times. So we figured out that it would be a good idea to get some sleep. So we went home at this stage. There was no game done, only the engine for the game and around it the game could be written.

Morning after

We returned to class 6218 at 8:30 am. We had now 1,5 hours of effective time to improve our creation. You were able to chose any ball that you want to hit. But really you just want to hit the white one. From

this we got ourselves nicely to the next problem. Which was tracking the balls and to be more specific which of the balls has gone outside of the table or game area. Here again the box2d showed it true itself. It doesn't identify bodies/elements whit anything. it just knows their location and that's it. When the balls bounce around it will get hard to get track of each balls location so we needed to find out a way to indentify balls. We talked earlier about the userdata and it was the result in this case too. But. The user data already contained the information about the image so we wasn't just able to set identification data to there. We created an array that included both the image and the identification data. So now we have both images and identification working.

Time was running out so we made a choice only track the white ball. We defined coordinates for every pouch and in update process checked if the white ball is inside of these coordinates. After that an alert pops up and tells that white ball is outside, but nothing else happens. We run out of time at this stage. We had started building the game logic and it will be relatively easy to create now around the engine.

After all we managed to create relatively realistic physics for ball movements and started the game itself. If we should conclude all this process to one sentence it would be this: "Note to self: DO NOT USE BOX2D WEB". Why? The box2d itself seems nice and does things nicely. But the fact that it lacks documentation, or manuals or anything that would describe even a small amount of the engine makes it useless. Otherwise it would be great. But now it's just like trying to learn french on my own without dictionary or any help. It is plausible but just too hard, and makes us want to commit suicide too much.