Game Development Journal

Ninja Ball

Group 7

Oscar Bogenberger	19275153
Elliot Cleary	19264259
Tomás Crowley	19263546
Dawid Kocik	19233116
Ronan McMorrow	19235208

Genre

In week 4 we had organised a brainstorm in order to find the most promising genre and the best idea for the game. We gathered in the ICT centre and every member of our group had come up with a different idea. We listed the main features of each game and everybody had drawn a sample game screen to help to visualise it.

These ideas consisted of a tower defense game, a puzzle platformer game and a game reminiscent of “Doodle Jump” or “Icy Tower”.

We discussed these ideas and voted, deciding on ninjaball as our project. This idea went through changes as well. Originally it would have been a stealth platformer with puzzle elements similar to “The Swindle” but this was then scrapped as we then decided to focus more on the platforming aspect of the game.

Theme

As we wanted to make a game focusing on pinball based platforming we decided on a pinball theme as it gave us a lot of interesting objects to work with. Try to get to the top of the tower avoiding obstacles.

Platform

The intended platform is Mobile. The controls are made to work best on a phone. The movement is too difficult to control using a mouse. We did not implement any keybinds to work with movement on PC.

Inspirations

1. Pinball
2. Super Meat Boy
3. Shovel Knight

How we wanted to set ourselves apart.

We needed to ensure that our platformer felt distinct and different to most others in the genre. To do this we had to not only develop our own ideas but also to look at what design features were used by our inspirations, and decide how these features could be changed and implemented together.

By doing this we landed on a few design principles we tried to follow in our game and level design.

1. Momentum based platformer

“Super Meat Boy” was the initial inspiration for our character’s movement. the character having a good sense of momentum while still feeling responsive was the feeling we were going for. However the bumpers in the game “space cadet pinball” were interesting as they would increase the speed of the ball drastically, this led us to the decision to use objects such as these to provide our character with the speed needed to make jumps and clear obstacles. We tried to follow this principle in the design of our objects and in the levels where the player would be tested in their ability to control the speed of these objects.

In week 4 we decided on this design principle while planning our game. and in week 5 we began to explore this principle in our level design drafts and in the design for our objects.

2. Compact level design

Due to our decision to use a mobile phone as our platform and the features of this system. we decided that compact levels would be able to provide an interesting challenge to players in this type of momentum based platforming, as the management of speed in a confined area would provide a different kind of challenge than most platformers.

In week 5 we began to work on this principle in our level design drafts.

3. Vertical platforming

We wanted to use the phone in a vertical orientation and this of course hugely impacted the design of our game. The main challenge of our levels was in moving upwards, not forwards.

In week 5 we began to work on this principle in our level design drafts.

What we wanted to do differently

We wanted to do a boss level at the end of the game but we didn't end up with enough time to implement such a level.

Early on we wanted to make the ball rotate as you move around but we realised that wasn't very feasible after trying it. Hitbox detection for jumping becomes easier to deal with if the ball’s rotation is fixed.

We originally had a crouch button to stop or slow down the character but we realised that we didn't need it as friction gives the same effect.

Level Design

We started level design in week 5 and we started creating the json files in week 10.

When looking for a way to make the levels in Corona we found the Tiled level design software on the corona website.

When designing the levels we wanted to make them challenging and have a decent rise in difficulty between each level.

When trying to get the levels to work in the game we decided to use ponytiled to convert the levels to usable code in Corona. However we had trouble with this so we decided to use the json files to decide where to place the objects in the levels.

Character Design

The design of the ninja ball within our game was basic yet efficient, we went with the old arcade style of game and thus wanted to give our own twist on it, so we made our character (ninjaball) into a perfect circle, this offered us a lot of opportunities to mess around with the character and a lot of challenges to overcome. We based our design off chinese warriors/ japanese ninjas and tried to make it look like modern day pixel art, there were discussions over which design would be best for our character but in week 5 we agreed on the one shown on the cover page.

Game Loop

Our game loop runs on every frame in the game. We used it to assure smooth acceleration of our character. We took that idea from the “Sticker Knight” template, which was available on CoronaLabs website. We had it replaced our previous character movement code in the middle of the development of the game.

Gameplay

The gameplay we tried to emulate in this game is that of fast paced platformers such as “Super Meat Boy”, but using a more compact level design than is typical is that genre. To achieve this we tweaked the characters controls so that the player would have a large amount of control over the movements of the ball while still retaining a sense of momentum and speed. we also had to take the level design into consideration, creating levels that would challenge the players ability in both precise control of the character and in their use of the environment and the objects in it.

Objects

1. Flipper

The flipper within the game was used to boost the ninja ball upon touching it in a straight upwards direction, giving the player an edge to get further in the level and ultimately move onwards to the next level.

2. Bumper

Bumpers were the wild card within the game, the ninja ball could hit off this at any point since the bumper was also a ball. thus allowing the bumper to boost the ball in the direction which the ball touched it, this could be used to the players advantage or disadvantage.

3. Spikes

Spikes were traps to kill the player if they were touched. The ninja ball would fade out and respawn in a suitable position on the map. In week 10 we added an option to respawn in the lower part of the level if a player died in the lower part of the level, and then respawning higher up if they made more progress.

4. Spring

The spring was used to propel the player into the next level. It was one of the last objects we worked on. At first we wanted to have it designed like in a pinball machine; a tube that the pinball gets shot up through. However we later decided the levels would start at the bottom and work their way up so having a tube was unnecessary.

Cut objects

These are objects we initially wanted to add but then decided against.

1. Saw blades

Saw blades would have killed the player on contact, same as the spikes but would utilise the kinematic physics object type in corona and would have moved in a set path.

2. Doors

Doors would have been used to lock off access to certain areas. they would also be kinematic objects moving up or down to barricade certain entrances.

3. Switches

Switches would have been objects used to interact with the map. the player would collide with them, flipping them and affect the movement of doors or saw blades.

Controls

Since we knew that our game was going to be released for the mobile platform, we knew that we should implement simple controls that will be easy to use. We went through a few different ideas, including having a ‘crouch’ button which would function similar to the “Minecraft” shift button, slowing speed and preventing the ball from falling off platforms in order to help with the more precise platforming sections, however we soon realised that this slowed down the pace of the game too much and made the controls feel too cramped.

Our final decision was to have 3 buttons: left, right and jump. During the early development stages we only had 3 grey rectangles that acted as buttons. In week 12 they were replaced by yellow arrows on red background. Each button had a slightly different shade of red, so the buttons would look like they were merged and we didn't have to separate them with unnecessary black lines that would take away from the visual quality of the game. We have implemented multitouch in our game as otherwise simultaneous jumping and movement would not be possible.

Character Movement

Character movement had changed a number of times since week 6 before we decided to move the main character by applying force on every frame of the game, rather than relying solely on touch events for working out the acceleration. We kept improving the movement and jumping till week 12.

Assets

Since we had a set theme for our game obtaining the assets also took some time. During week 5 we were looking for pinball-like sound effects, graphics for spikes and other objects. We had to make sure that graphics were not copyrighted which presented additional challenges in finding the game assets. We couldn't find good quality textures for all the obstacles, so we had to design some of our graphics ourselves. The background graphics for each scene were found mainly during week 10.

UI Design

UI Design started with creating a scene with level selection. Our early concept art only had the movement buttons as we worked on the game more we realised that we could use 3 buttons similar to game controls for navigation between different menus.

In week 9 added credits, and Ronan and Oscar worked on the text informing where we downloaded some of our assets from. We also realised that we need a pause button with a menu in the form of an overlay. During week 10 we had a separate mute button in the form of an icon but in the final weeks we removed it in favour of a descriptive mute/unmute button with text, which meant that a player could mute a game from any screen.

Communication challenges

At the beginning of this project we could communicate freely among our team since we would see each other almost every day during the week. When the university closed down we had to adapt to this new situation, where meating in the ICT centre was no longer possible. We already had a Github repository set up, so the file management was slightly easier. However, we needed an additional way to communicate, which would allow us to discuss ideas and track our progress on a daily basis. We decided to use Discord, which is software that allows you to create your own server, chat through text, call and share small files online. We decided to use discord because every member of the team had it and we already had our server set up. We have encountered some problems during week 11 when our game apk was too big to fit into the 8 megabyte limit for file size. Luckily, Elliot had Discord Nitro, which is a premium subscription for this software allowing you to upload large files. We could send an apk to Elliot through email and he could upload it to our chat server.

Programming challenges

1. Flipper - The flipper gave us a lot of problems in the beginning. Rotation around a point is not a method within lua. We attempted to create our own code to make lua rotate the flipper around the yellow bolt in it but the code proved too difficult to get working. Elliot tried to use joints and a separate object for rotation about its center, but this didn't work either. By default lua will try to rotate a body around its centre so Dawid found a workaround where we extended the hitbox of the flipper to be (almost) double the flipper's actual length. This way the centre of the flipper actually would be the yellow bolt and we could rotate it around the point we wanted without any extra code.

2. Character Movement - We took multiple attempts at creating a reliable piece of code for the character movement. In the early weeks of programming the game we had it moving in a linear fashion. Later, we tried to add manual acceleration and deceleration, however the character felt unresponsive. The breakthrough happened when Tom found a function for swift character movement in the Sticker Knight template source code. We implemented that in our game and later customized it e.g. Tom expanded the code further and added deceleration.

3. Level Design - The work on level design began from the very outset of our work on this game. Oscar designed the levels in Tiled, which is a level generator software recommended for creating levels for games in lua. However a problem arose as it required ponytiled, which is a special map loader. We took many attempts to get that working, including creating each type of object as a separate module. We could not get it to work with our version of Tiled (values for objects group id greater than 1 billion, incompatible features, issues with absolute paths, unsupported texture file) and that caused a delay around week 11, so we created a simplified version of json files and Dawid coded up our own map loader from scratch.

4. Level Transition - Implementing the level transition was also a bit challenging. The idea was simple as after hopping onto a spring the ninja ball would be fired off into the next level. We worked on the level transition in the later stages of programming the game. We tried to reload our game.lua file with an update to level variable, however the scene considered itself as 'already created' and thus would not reload. To overcome this, we created an invisible callback scene, which returns to the game scene. Also, our home-made map generator would not always be run or physics of old objects would be retained or errors

would crop up. Later, we could not pause the physics engine as according to Corona this was impossible during a collision with a spring. We restructured, rearranged code, performed some actions with delay and added extra functions to make sure all these errors are fixed.

Compromises that had to be made

We didn't make any boss level in the end as we wanted to make sure that everything we had already done was working properly. That includes the controls, the movement and transition between levels. We thought that it would be better to clean up what we had already made then leave the game in an almost finished state just so we can have a boss level. Our original design for the boss level also made heavy use of switches, doors and sawblades, which were objects we had plans for but didn't make the cut. the boss level would have had to be drastically redesigned in order to work and so we decided our effort would be better spent elsewhere.

What we learned

Design and development of “Ninja Ball” let us gain an insight into a programming language other than Java. We learned how to design and make a game that will be consistent, challenging and enjoyable for the players. We have improved our organisation, teamwork and time management skills. We also learned not to assume that a third party software will work correctly. Every member of our group has put a great effort into this project and we hope to keep on expanding our game in the near future. We learned how difficult it can be to remove bugs we encounter. Our game was on a pretty small scale and we didn't manage to make it completely bug free in the time we had so we can imagine how frustrating it can be to try to make a larger game bug free. We have a newfound respect for game developers.