Shodo Post-Mortem

James D. Cross

Intro

Shodo is an arcade style brawler where you play as a ninja who must fight his way through an evil lord's palace to defeat him. Aside from the standard punches and kicks indicative of a brawler the main fighting mechanic consists of combos. An example would be the spinning kick combo initiated by pressing the sequence BBA on the controller.

Originality

There are very few top down brawler games, they are usually side scrolling. This makes the design of the game more unique. The line on paper art style reflecting the calligraphy style of japan (Shodo is the Japanese word for calligraphy) gives the game a fresh look. Introducing combos usually used in side scrolling fighting games as the main mechanic in a brawler is a relatively unique idea.

Original concept

Shibo comes from a village where becoming a sumo wrestler is a great honour. His glory was taken from him by lord Sukinī (Japanese for skinny) who wanted his son to be the village champion. Shibo (Japanese for fat) vowed revenge and went to the lord's palace to exact it. This story defined the design of the levels, Shibo ventured through the palace to get to the final boss stage (the lords chamber). It also was used in the game mechanics as Shibo would enter the "Sumo state".

This lead to more powerful combos and attacks. In one of these he essentially became a ball of fat and crushed enemies. This was abandoned when it became clear that the graphical art style had a more serious tone. The originally planned pixel art would have better suited this comedy element. Also there was no need for this story hook as the aesthetics of the game became the hook. However the level design and basic plot of Shibo fighting through a palace to defeat a lord remain.

5 Things That Went Right

Graphics

The line on paper graphics style of the game was achieved through the use of a "green screen" or chromakey effect. A shader was written for Unity that replaced the white in the images we rendered out from blender, with a background image in Unity. This meant only the black lines (created by blenders outline shader) showed through. To make this work effectively we set the skybox in our project to an image of paper. This meant that the image would always be present in every scene and always be behind every sprite. This was a great lesson in the power of shaders and the huge difference they can make to the look of a game.

Animation

Animation was always going to be a problem for our game as we needed many animations for the different combos and enemies (in eight directions) because of the top down perspective. Fortunately Reuben has some skill using blender and he discovered Mixamo. This is a site the lets you download animations for use with 3d models for free. By rendering out 3d models to 2d images we could make 30 frames per second animations in 2d (in eight directions). This could be achieved relatively quickly once the initial 3d models where built.

I looked into implementing animations in Unity and decided that blend trees where the best approach. They allowed us to keep animators simple with fewer transitions. Blend trees also allowed quick animation of characters from 8 directions. By using the getAxisRaw variables (1,0,-1) for each direction we could copy the blendtrees and plug in different animations for each game entity. I would definitely use this approach for any 2d game I make in the future.

Enemy AI

Unity's navmesh system was used for the enemy Ai. (the navmesh system uses the A* algorithm to calculate the shortest path on the navmesh). The navmesh is usually used in a 3d space however it can be used in 2d. Unity takes in the 2d points of movement, then moves the agent in a 3d space then converts the agents position back to 2d. So as far as the agent is concerned it is moving in 3d. Static colliders are used to create a boundary for the mesh. The colliders combined create a single temporary object which Unity then destroys once the navmesh is created. This allows the enemies to follow the player in a defined play space.

The navmeshagent2d radius feature allowed us to keep the enemies a certain distance away from each other and the player which made combat more realistic. The enemies originally just ran away when there health was low. We later adjusted this so they run back after 5 secs with more health. This created a new gameplay element, if you don't kill them while they are running you will have to deal with them later at full health. We also changed this as it looked like a bug when they simply ran away and got stuck on a wall still running. The navmesh system built into Unity has many advantages and is the easiest way to create convincing enemy AI.

Combo system

We decided early on to use a controller for our game as it would give more fluid control than the keyboard moving in eight directions. However because the game is to be built for WebGL we did provide keyboard control as well. We used the Unity plugin inControl ,inControl is an input manager for Unity3D that standardizes input mappings across platforms for common controllers. This way we knew we would not have trouble with input on different systems. Troy also worked on the combo system. Originally the combos required buttons to be pressed within a range of 0.2 and 0.5 seconds. One thing we discovered from testing was that people where finding this too difficult. We adjusted the combo system to a range between 0.2s and 2s. This makes combos much easier to execute which means the player can start using the main combat mechanic straight away. It also had the unexpected bonus that you can "store up" a combo sequence (say A, A) then run in to an enemy and execute it (B).

Intro cut scene and QTE

A late addition to the game by Albert was the intro cut scene and the quick time event system. Both of these where made possible by Fungus. Fungus is an open source tool for creating interactive storytelling games in Unity 3D. Fungus allows you to lay out cut scenes in one Unity scene. One cut scene flows to the next using simple visual scripting. This made introducing this feature a lot easier than it would be to try to do this in Unity natively. The timing and display of text and images is also greatly simplified by use of this plugin. It is also highly adaptable which let us use it for the QTE system. This taught us a valuable lesson that one of the best features of Unity is what you can add into it. The asset system provides a lot of flexibility and saves time on a game project.

5 Things That Went Wrong

How Graphics Affected the game

We committed to a high workload early in the project because of the art style we went with. This pushed the whole project forward as the outlay at the beginning took so long. Deciding on using eight directions for each sprite may have also been a mistake. This doubled the work load (as opposed to four directions of movement). Rendering and importing sprites into Unity took twice as long and was a very manual process. It also meant double the amount of animations had to be created for each character in the game. We had to use PNGs because we needed the alpha channel to generate the Chroma key effect for the art. PNGs are a larger format than jpg for instance and we never looked into compression. The sprites also had to be 500 x 300 resolution as they rendered from Blender to look good in game. The animations also had around 20 frames per direction. This meant that the project is around 2.55 GB in size which lead to problems such as building for WebGL taking 40 minutes which made debugging very time consuming.

This could have been avoided by early planning. If sprite resolution size is a power of 2 they can be DXT5 compressed in Unity. We also should have cleaned up unnecessary files more as we worked on the project to avoid bloat. The animations also caused a time sink. We spent a lot of time getting sounds and events to trigger in the right place because the players fist would only hit the enemy in the middle of their punch animation not at the end. Unity does not have an animation frame access method you can only access a time in the animation which is not ideal and trigger methods in animations is limited only to certain methods on an object.

Testing

Delays in the project meant our alpha only had two levels and two enemy types. Our Beta was non-existent we just had to test the alpha again. Consequently play testing was not really used, we only had the final game made in the last week because we spent so much time on the visuals. So we never really got to test the essential question “is the game fun?” We may think it is fun but without outside opinion it is impossible to really know. The game also suffered in design because everything was pushed forward and we had no feedback. The level layout and enemy difficulty are not balanced as well as they could have been. In the next project I would definitely either find a way to automate our blend trees being populated from blender or just do the whole project in 3D. This would lead to a more polished game with more time for development.

Development

The structure of game development could have been improved. We should have made a single level that just spawned enemies and rigorously tested the fighting mechanics as this is essentially the whole game. Although the levels where designed and tested we didn’t iterate enough. We should have created graphically basic levels with simple cubes (no animations) representing the player and enemies. Then tested the level flow and the pickup placement as we were developing systems. If the game was fun to play this way, it would of guaranteed that when the graphics where introduced it would only enhance the core game. Consequently our game lacks depth. The gameplay is not varied; the enemies all have essentially the same attack except for the main boss at the end. There are no different skills or tactics required by the player. All combos are available straight away meaning the player has no new skills to learn as the game progresses. When we redesigned the levels we introduced a large balcony this was a bad idea because when enemies die the bodies sit off the balcony ruining the illusion of 3D space. This meant a lot of fiddling with baking navmeshes as the enemies slide around when you are fighting them. This could have been avoided through testing out the levels with basic graphics.

Sound

I worked on the sound design for the game and composed the music (except the theme in the intro cut scene that Albert sourced). I was fairly happy with the main game loop for the music. However I should have concentrated more on sourcing different sounds for player and enemy attacks. Only the player makes a sound when he attacks the enemies, the enemies only make a sound when they are hit. I feel this hurts the immersion and impact the game could have had. There are no sounds in the menu either or when the player completes a combo, these where not sourced due to time restraints. I held off on composing music in the beginning because I was waiting to see a more complete version of the game before I got stuck in. This was a mistake I should have simply composed a series of loops with different qualities and then I could have used that sound bank to quickly sketch out the music later in the project. The same applies to enemy noises I should have made a large sound bank in the beginning for later use.

Polish

We never had time to polish the game. Hence there are still graphical errors such as blood particles clipping from the 3D floor (the particles move in 3d space under the 2D graphics) this produces a weird effect where blood looks like lines not drops. The damage text for the enemies is still displayed in the death screen. There are not enough visual effects in the game. In most fighting games there is some kind of visual effect when you pull off a special move other than just the player kicking. We experimented with introducing colour in to the game using water colour paints. This did not look how we wanted so we abandoned it because of time constraints. This combined with the lack of sound variations takes away a lot of excitement (arcade feel) the game could have had.

You may notice that the perspective of the levels in the game changes. This worked out ok but was due to the fact that we were still deciding on the angle to use and before it was decided, we had run out of time. The game would be more unified with a uniform perspective. We had to cut the amount of combo's which would have made combat more interesting. All the enemies look the same as the player. This to me is almost game breaking, it stops the game from being professional and leaves it in the realm of a great prototype. The original 5 enemies designed for the game where also not implemented which hurts the depth of the game. They had different movement speeds and tactics and would have made combat more interesting.

The level layout went through 3 iterations this seems like a natural progression for a game to take however it may have weakened the game as a whole. In the original design we had a story that related to the level layout of the game and related to the gameplay (The player had a "Sumostate" with more powerful combos) now we have a very generic story which is essentially the just the basic elements of the original story. In future I would try to get the foundation and tools in place to streamline our workflow so we spent less time manually entering animations and re-creating prefabs (work was lost when someone applied a change to a prefab and it effected all of the prefabs of that kind in the game) so we would have time to polish the project.