ROTAT-A-BOT

Game Analysis

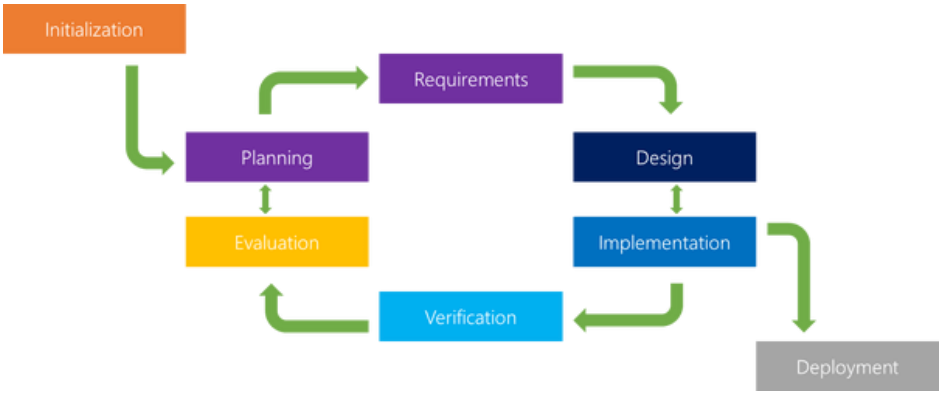
Daniel Cutajar & Dante Camenzuli

7/6/2019

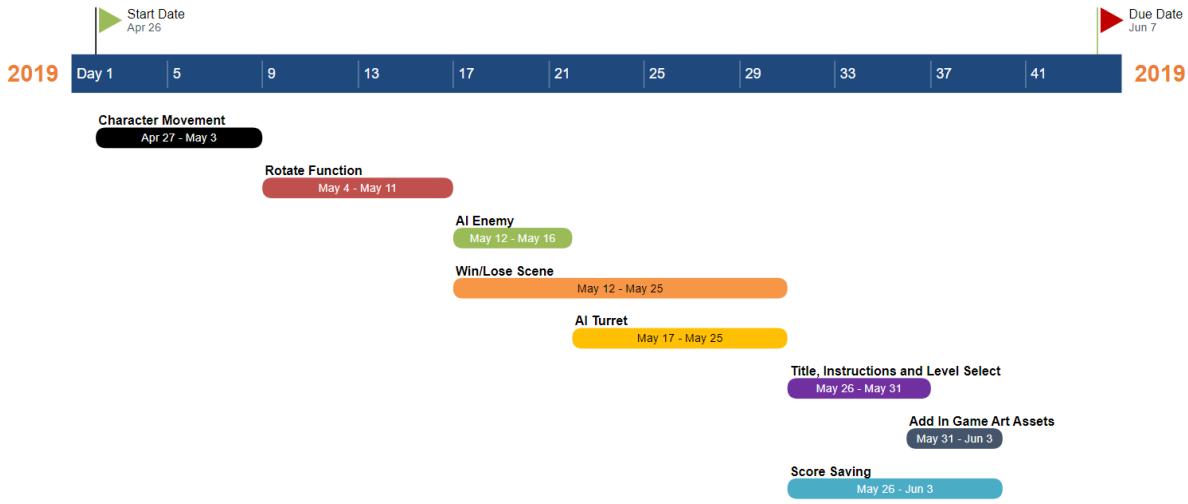
PROJECT TIMELINE & SOFTWARE DEVELOPMENT MODEL

Without any fixed tutorial to follow for design and programming, we are able to follow our own software development model as precisely as we desire. The model we shall follow is the Iterative Model where we imagine the finished section we desire to create. We then deduce the requirements to make that section, code it and test it to see if it functions accordingly. Should any errors occur the cycle from planning begins again to find a new approach. If it succeeds the planning of the next iteration step begins.

Our freedom will allow us to stick to this style and create a solid and functional workflow. Having full control will make us be able to work at our own pace and assess every component we add in and make sure it is up to standard as it will affect the future iterations.

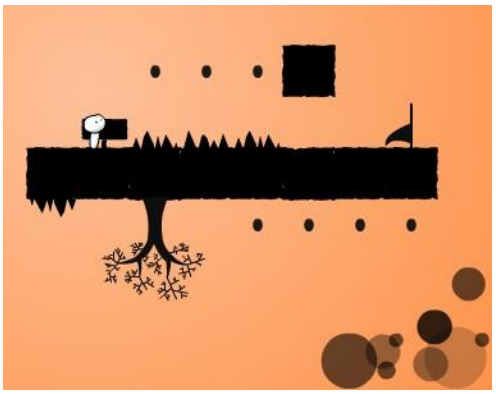


A project time map was drafted with when we desired to get each component finished. Sticking to the time map will guarantee a successful product and allow us time to build and pass the game out to a few players for testing and feedback. Components were ordered in a way that no component that requires a previous component comes before it. Everything being made will be used in the future components.



INSPIRATIONS

The first order of business was to try to find a game we can emulate. We searched though play-throughs and looked up mobile games as well as indie games to find something that isn't overbearing in mechanics. We did find a rather simple game called *Topsy Turvy*, in which the main and sole character of the game – a simple humanoid silhouette named Topsy – must travel along the landscape to reach the flag. The whole level flips at 90-degree angles, allowing the silhouette to reach safe places to rest or land after a jump, and with each new level, the craftier the puzzle becomes, making the game increasingly difficult. Another game that followed this mechanic was *Rotate*. It functions in the same way except that the rotation of the level is up to the player pressing the C and V buttons for directional rotation. This is the idea we have in mind for development rather than *Topsy Turvy*'s rotation on corner contact.



COMPONENTS

The level rotating mechanic is main selling point of the game which we've decided to adopt. The design style is up to the graphics artists' discretion.

To run this, we would require a script to not only move the player but rotate the level around the player when turning a corner. Another script to dictate any enemy movement, primarily, tracking the player around the level: at least one level will require an enemy character moving towards the player, and another having a turret that shoots at the player when near. We'll also need a scoring system of some kind – probably time-based – and a game manager script to run the game smoothly.

PLAYER MOVEMENT

The game concept is that of a 2D platformer with the rotation function being used to move around the puzzles. This would require the player to have classic player movement with the arrow keys moving left and right, as well as the jump button allowing them to jump. Collider boxes will be needed to check the ceiling and floor to prevent constant jumping up into the sky.

For easier design and less game issues plans to remove the ability to rotate when jumping will be added. This will be done using the jump collider checking if the player is grounded to allow jumping again. If the player is not grounded they may not rotate. Similarly, player movement will be frozen as well as gravity when rotating for the player to map out what they need to aim for when the map is rotating, then going for it after the map is locked in place.

MAP

After getting advice from the instructor we decided the best course of action from a development aspect and a game art aspect is to create maps using tile maps. The game art section would make unique tiles that we would paint around to create the map. This lessens the burden on them and allows easy map fixing should something not work.

MAP ROTATION

The map rotation is the key concept and needs to function properly. We debated whether to rotate the map or the player settling on rotating the map to be much easier. This requires a function that would rotate the map by 90 degrees exactly in the direction desired. For smooth gameplay added requirements include the pivot of rotation being the players position so they never change their location in the level.

CAMERA

The camera will follow the player keeping them in the centre of the screen. Rather than taking player coordinates and following the easier solution is to make the camera a child of the player so that wherever the position of the player the camera will always be at x = 0 and y = 0.

BATTERIES + KEY + DOOR

Batteries will appear on the map along with a key. The batteries provide points and the key is needed to unlock the door. When the key is collided with a check will be switched on allowing the player to collide with the door. The batteries colliders will cause it to destroy on impact with the player and also increase the score counter. (Concept in progress).

ENEMY AI

Plans are for two types of AI. A moving AI that cannot jump. This AI takes the players position relative to their own. Should their distance be less than 4 then they will approach the player via their capable movement (horizontal moving). If more they will remain idle. The turret AI will track the players location shooting a constant beam at them. This will be manipulated to activate a button of some sort for level completion. Should the player be out of range then the turret still aims but does not shoot.

MODELS

Some things we've all agreed on is that we would have five distinct levels based around the Chinese elements: Fire, Water, Air, Wood, and Metal. Each element a different world with different challenges, landscapes, and characters. (Concept in progress). The models will be left up to the game art students with specifications of each asset being 16x16 to fit into the tile maps.

GAME UI

This will be handled through various scenes. A title screen which will connect to the levels and instructions. When a level is selected that level's scene is loaded and played through. Whether you win or lose you are taken to the level complete scene which will show UI based on whether you won or lost.