**Game Development Patterns**

***Assignment 1***

*Luke Courtney*

*K00267879*

**Introduction**

The objective of assignment 1 of Game Development patterns was to modify the Thomas was Late game in such a way to turn it into a platformer game similar to that of Super Mario or Sonic the Hedgehog. We were to include at least three levels, three types of pickup or collectible, and three types of enemy. The split screen mechanic from the original Thomas was Late game was to be removed, as was the second playable character, Bob.

**Character Changes**

After removing Bob, the second playable character, and the split screen function, I began working on an update to the movement system. The movement in the original Thomas was Late game is, in my opinion, too responsive. This means that when you press a button to move in a direction, the character is immediately moving at full speed in that direction. I sought to resolve this by adding an inertia system to the game.

The inertia system that I added allows some time for both the player and enemies to get up to speed when they begin moving, and some time to slow down to a stop when they stop moving. This adds a sense of weight to the characters, and in my opinion makes the platforming more engaging, as the player now has to consider the stopping time when trying to land on a small platform, or the distance required to get up to full speed when running for a jump. Over the course of the game, the player is able to raise the maximum speed allowed for the player character through the use of pickups. This does not however increase the acceleration of the character. As such the system rewards the player for being able to keep moving in one direction long enough to build up more speed.

A computer screen shot of a program code

Description automatically generated

Figure - Inertia code for right movement

As shown in Figure 1, the inertia code is relatively brief. The character has a set of values defined in PlayableCharacter.h that set the acceleration, deceleration, current left speed, and current right speed. In the example shown above, if the character is moving to the right and their right speed is less than the maximum speed allowed, their rightward speed is increased. If they’re already at the maximum speed, the rightward speed is not increased. The same process is repeated for left movement. The left speed is then subtracted from the right speed, and the X value of the character position is updated by the resulting amount.

Speed is preserved when the player respawns after dying. This further encourages the player to keep moving forward as fluidly as they can, not stopping to consider jumps or other obstacles, and instead take risks knowing that if they do fail to avoid an obstacle, they will likely be able to get back to that point of the level relatively quickly with their preserved speed. This system was inspired by games such as Hotline Miami where death is not game over, and being able to restart and get back into the action quickly is a core mechanic used to keep the player engaged even when they fail at a level several times over.

**Enemies**

****

Figure - Enemies

The original Thomas was Late game does not contain enemies, with the challenge instead being to get the two playable characters to the finish line. In the game that I created for this assignment, there are three types of enemies. These are white, red, and blue enemies. To create them I repurposed the Bob character from the original game. Each of the three enemies has a different behaviour.

* + The white enemy is the basic type. It moves left and right along the ground around what I call a “patrol point”. This is a point given to the enemy around which it moves. The enemy initially moves to the right, until it reaches a certain distance from the patrol point, at which point it turns back and moves the opposite direction until again reaching a set range from the patrol point. The enemy additionally has a condition to turn back if it hits a block and cannot move further, as otherwise it would be prevented from moving to the desired range before turning back.
  + The red enemy is a more difficult variant of the white enemy. Initially it moves just like the white enemy, but as the player moves it will start jumping. As long as the player is moving, the red enemy will continue to jump. This forces the player to time their own jump just right to make it over the red enemy when it is at the bottom of its own. These enemies are significantly more difficult than the white enemy as their jumping behaviour is only apparent when the player attempts to move past them.
  + The blue enemy is the most unpredictable of the three. Similar to the first two, it moves using the patrol point system. It is also set to continuously jump, making it more difficult to avoid. The unpredictable nature comes from how it changes direction. The blue enemy will not just change direction when far enough from its patrol point, but rather when its far enough from its patrol point and the player is moving. This means that if the player stands still, or stops moving for any reason, the blue enemy may continue to move further than expected and end up in a part the level that the player did not expect them to be in. This can allow the enemy to catch the player off guard, and reward players for continuing to move at all times.

A computer screen shot of a code

Description automatically generated

Figure - Enemy patrol code

As shown above in Figure 3, the enemy movement direction is decided using the Boolean variable “Flipped”. If the enemy is not flipped, and is more than 150 pixels to the right of the patrol point, they will be flipped. When the enemy movement is handled, they are moved either left or right depending on if they are flipped or not. Non-flipped movement is to the right, and flipped is to the left. All enemies respawn if they fall out of bounds or hit an obstacle. There are three enemies in each level.

**Level changes**

There are three levels in the game. Each level is significantly different than the previous level, forcing the player to take a different approach to each. The first level is relatively short but has a difficult obstacle near the end that will likely require several attempts. The second level is not as difficult, though still has its own challenges. It is much longer than the first level, and rewards speed. The third level is slightly shorter than the second but focuses on precise platforming to make it through.

The sprites and background texture were changed in order to bring a new style to the game. The background texture is now a blue sky, and the main ground sprite is grass. The lava obstacles have been replaced with large metal spikes, and now kill the player on contact, rather than waiting for the player to sink in. As the player character is taller than one tile, the previous system allowed them to walk through any obstacle that was one tile deep.

**Pickups**

****

Figure - Pickups

Three pickups have been added to the game. These pickups, when collected, give the player a bonus or upgrade to help them complete the level. When collected, the pickups are removed. Each pickup can be collected once per level. The pickups alter the players speed, gravity and time remaining in the level respectively.

A computer screen with text and numbers

Description automatically generated

Figure - Speed pickup code

The speed pickup, represented by a yellow star raises the player’s maximum speed. As mentioned earlier, the players acceleration remains untouched. This encourages the player to keep moving in one direction for longer in order to build up more speed and get closer to the current maximum.

A computer screen with white and blue text

Description automatically generated

Figure - Gravity pickup code

The gravity pickup reduces the effect of gravity on the player, allowing them to jump higher. This can make more difficult jumps easier and can even open up new paths through a level that would otherwise be impossible. The ability to stay in the air longer works well in combination with the raised maximum speed from the speed pickup, as it allows the player to avoid obstacles for longer and build up even more speed.

A computer screen with white and blue text

Description automatically generated

Figure - Time pickup code

The time pickup grants the player additional time to complete the level when collected. This can allow the player extra time to make it past any obstacles they are finding particularly difficult, or to try out new paths through the level that they may not otherwise want to risk time on.

**Conclusion**

Overall, I believe the changes made to the original Thomas was Late game do a good job at changing the gameplay into a platformer in the style of Super Mario or Sonic the Hedgehog. I gave the game a focus on speed and trial-and-error gameplay inspired by games such as Hotline Miami. I feel this gives the game a less often seen style of gameplay and keeps it engaging even through repeated failures.