Santa Jump

Brent Cahill,[1, a)] Lukas Ekberg,[1, b)] and Ryan O'Kane[1, c)]
School of Computing and Information Systems, University of Melbourne

(Dated: 20 October, 2018)

We present a look at the extended design and implementation of the simple Unity-based game "Santa Jump."

I. INTRODUCTION

"Santa Jump" is a simple Unity-based game in which the user controls a player that jumps from platform to platform, attempting to improve upon previous high scores. The creators used a combination of made-from-scratch and purchased assets in order to develop the correct feel for the game, and allow them to focus on the important assets like scripting, shaders and lighting calculations, and controls and gameplay.

In the end, the final game ended up being visually pleasant; reminiscent of a Christmas themed pack for a larger game, with controls that were "intuitive" and "felt natural." Although some aspects of the game were a little bit trickier to understand, nearly everyone that participated in beta testing felt comfortable with the game after only a few tries.

II. THE GAME

A. Gameplay

The game begins with a character jumping on a platform. The objective of the game is then to control this character using the arrow or WASD keys to achieve the highest possible score, which corresponds to maximum player height. The player can increase their height by jumping from lower platforms to higher ones continually until missing a platform, causing them to fall out of the frame of the screen, ending the game. In addition to the platforms, "boosts" exist that launch the character significantly higher than simply landing on a platform. In the lower right hand corner there is a scoreboard that allows the player to see the performance of their current game run. In the lower left hand corner there is a boost gauge that tells the player how much of the boost is left when they have one. In testing, subjects who played the game said that the gameplay was easy to understand, and required little, if any, explanation from us after the fact.

[a)] Electronic mail: bcahill@student.unimelb.edu.au
[b)] Electronic mail: sekberg@student.unimelb.edu.au
[c)] Electronic mail: rokane@student.unimelb.edu.au

B. The Game Look

The overall goal for the look and feel of the game was a Christmas theme. To achieve this, free-to-use assets, available on the Unity Store, were utilised for the player and backdrop. In addition, to keep the feel of the game lighthearted, a Cel Shader was implemented on the character, but a Phong Shader was used on the platforms to provide a sense of contrast that increased the overall "Toon" feel of the game.

C. Scripting

The character was moved by using the left/right arrow keys, or the "a"/"d" keys on the keypad. Each of these keystrokes applied forces to the character's rigidbody, creating a sliding effect when combined with the properties of inertia. The character's jumping animation was taken from the Sleek Toon bot asset, made by Seth Ward, and available free on the Unity asset store. When the character reached the edge of the screen, it was "teleported" to the other side using `transform.position`.

As the character was constantly moving, the camera had to follow so that the player could see the character they were controlling. To achieve this, a `CameraScript` was attached to the camera that tested to see if the character had collided with a platform that was higher than the camera. If so, the camera would smoothly move up to that new platform's height. Otherwise, if the player got a boost and was moving up without the use of the platforms, then the camera would track along the player's height.

The random generation of both platforms and boosts were left to the instantiation of prefabs. The boosts were created once every 10 seconds within the game, with a height approximately equal to the player, with some randomness for both x and y values. The platforms were generated at every 4 height values, with random x values. These were generated every time that the character increased his height by 4, and were created above the highest current platform, so that there would always be platforms on the screen. Initially, 5 platforms were instantiated so that the game could begin. All platforms and boosts were destroyed once they exited the game screen, using `onBecameInvisible()`. In addition, only a small area of the background was ever rendered, and it simply shifted up whenever the player moved. All of this ensured that the game ran smoothly, without unnecessary rendering, optimizing the graphics pipeline.

The platforms were scripted such that any collision would only be registered if it came from above. This allowed the character to pass through the platforms from the bottom, which was essential to playing the game.

D. Shaders

As the Phong shader has been taught throughout the subject, more focus will be put on the explanation of how the Cel Shader works. Cel Shading is a method of shading that makes an object appear as flat, rather than three dimensional. This helps the object look as though it is a comic, or drawn animation, giving the game or video a distinctive feel. Cel Shading is achieved by ignoring specular highlights, and focusing only on the ambient and diffuse components of reflection. Rather than the gradient from lighter to darker areas in the shader, there are predefined boundaries within which the amount of light will all be identical. Depending on the Cel Threshold, there will only be a certain number of shading values. These values are created by calculating normal, smooth values for each pixel, which are then quantized down to significantly fewer shading values depending on the threshold. These quantized pixel values are then the final colors that will be rendered. This gives the final image a "flat" look, as the borders between differently quantized pixels will be extremely visible and pronounced.

Although Cel Shading often utilises edge-detection or backwards culling to create black outlines for objects, this was not utilized in "Santa Jump," as the combination of character material and backdrop already gave us a sufficient enough outline to make the character "pop."

III. QUERYING AND OBSERVATIONAL TESTING

The game was presented to six individuals of varying demographics. The demographics of these test subjects can be found in Table I. The overwhelming majority of test subjects stated that the controls were "intuitive" and "felt natural." Some of the subjects originally felt that the game was too easy, as such, the platforms were spread out more to make the game more difficult, and to show the subjects that the teleportation from one side to the other existed. Some of the subjects did not realize that going to the extreme edge of the screen caused the character to teleport to the other side, and were therefore surprised when it happened or when they were informed of this. In addition, some of the subjects failed to realize that there was a scoreboard in the bottom right hand side of the screen. In all, the subjects felt satisfied that the game was well fleshed-out, with intuitive gameplay and natural controls. They enjoyed the Christmas theme, as said that the graphics felt as though they were chosen with thought and care to preserve that theme.

Nationality	Age	Gender	Degree Program / Employment
Australian	25	Male	Masters of Geology
Australian	18	Female	Bachelor of Arts
American	20	Male	Bachelor of Science
Irish	23	Female	Fitness Instructor
Dutch	20	Male	Bachelor of Arts
Dutch	23	Male	Masters of Real Estate Management

TABLE I: Demographics of Test Subjects

IV. RESOURCES

The assets for the character and jumping animation were taken from the asset store, with URLs as below, respectively:

`https://assetstore.unity.com/packages/3d/characters/free-santa-claus-106193`

`https://assetstore.unity.com/packages/3d/characters/robots/sleek-toon-bot-free-34490`

The Phong and Cel Shaders utilise modified code from the following sources:

`http://janhalozan.com/2017/08/12/phong-shader/`

`http://www.shaderslab.com/demo-77---cel-shading-(formula).html`

V. CONTRIBUTIONS

Brent implemented the code for both the Cel and Phong shaders. In addition, he contributed to the random generation of platforms, as well as wrote and formatted the report as it now stands.

Lukas provided all of the player scripting, including movement and interaction with objects, platforms, and boundaries. In addition, he scripted the camera movement, added the "Game Over" state with replay button, and found and implemented the Santa Claus Asset with animation.

Ryan implemented the particle system for boosts, boost spawning and boost scripting. In addition, he implemented the random generation of platforms, the destruction of objects which were outside the game screen, and an infinitely scrolling background. He also created the heads up display with boost indicator and current score, as well as the main menu.