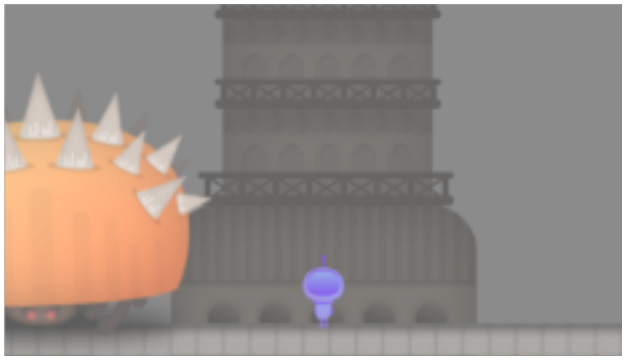
A 2D action side scrolling platformer for the Apple TV



Cal Poly Senior Project

Advisor: Zoe Wood

Winter - Spring 2016

Table of Contents

Abstract

Micronaut is a 2D side scrolling platformer video game application developed for the Apple TV with the tvOS SDK version 9.1. It explores human-computer interaction of platformer video games on the Apple TV and aims to give the player the most control to maximize enjoyment. Micronaut features five engaging levels with each level designed to focus on teaching the player the mechanics one at a time. Micronaut has been submitted and is undergoing submission for the Apple TV App Store at the time of this paper. Christopher Williams managed the software development and design, and Jacob Johannesen created the graphical assets.

Introduction

Apple released the 4th generation of its Apple TV in October of 2015 which features an App Store and a new remote. The new Apple TV remote comes with a touch surface on the top half with four buttons and a rocker (as shown below).

Figure 1: The remote for the Apple TV 4th generation. It features a touch surface on the top half.

This reinvention of the Apple TV provides a unique platform to build interesting games with very specific restrictions on controls and user interaction. However, when the new Apple TV launched very few games varied outside the simplistic control scheme of tapping to perform a single

action or the games require a third party controller to get the best experience. Micronaut was conceived to be an exploratory platforming video game that tries to give the user more control by utilizing more of the remote's technologies.

Micronaut is a 2D action side scrolling platformer developed for the Apple TV version 9.1 that was developed with the tvOS SDK with Xcode on a Macbook Pro. It features five levels, each of which features a certain aspect of the controls or challenges the player to utilize those controls effectively. This document outlines the development process including the decisions and challenges that have lead the game to become what it is today.

Jacob Johannesen provided the visual assets while Christopher Williams developed the application with tvOS SDK version 9.1 in Xcode version 7.3.

Related Work

The Apple App Store contained over 500k gaming applications back in September of 2012.[1] It is important to analyze these games to see what was done well and what should be avoided. Most of Micronaut's inspiration is derived from excellent games from various systems. The following subsections highlight the biggest inspirations that lead Micronaut's development.

Rayman Adventures

Figure 2: The title screen of Rayman Adventures for the Apple TV.

Rayman Adventures is an action side scrolling platformer and one of Apple TV's flagship games. The player controls their avatar by swiping left or right to run in that direction, swiping up to jump, swiping down to roll or dive, and swiping in the same direction the avatar is running to attack. This scheme gives the player a fine-tuned sense of control and capability to explore the levels. This sense of control and capability to explore are two important concepts that set Rayman Adventures apart from many other games on the Apple TV.

Figure 3: In Rayman Adventures, the avatar runs either right or left when the player swipes right or left and the avatar jumps when the player swipes up.

Micronaut was conceived with the notion to give the player control and the capability to move at their own pace, similar to Rayman Adventures. Thusly, Micronaut's control scheme is heavily influenced by Rayman Adventures. In both games, the player swipes right or left to run in that direction and swipes up to jump. However, to give the player a better sense of control and pacing Micronaut allows the player to swipe down to stop their momentum. Micronaut's design paradigms of exploration and control were heavily influenced by Rayman Adventure's control scheme.

Rayman Adventures may be downloaded to play on the Apple TV for free at https://itunes.apple.com/us/app/rayman-adventures/id1043589663?mt=8.

Super Meat Boy

Figure 4: The title screen of Super Meat Boy for the Xbox 360. Super Meat Boy is also available for various other platforms.

Super Meat Boy is an action platformer that is most known for it's difficulty and fast pace. However, despite its difficulty many people enjoy the challenge and are willing to try the same level over and over again. This is due to multiple factors, but the most important is the quick respawn time. The player is not punished too hard when they die, but are instead thrown right back into the action at the beginning of the stage. This keeps the game fast-paced and energetic so the player is excited and willing to tackle the challenge again.

Figure 5: Whenever the player dies in Super Meat Boy, they are thrown right back in the action.

Micronaut aims to keep the interest of the player by using a quick respawn time. Instead of using a cinematic approach to communicate the death to the player Micronaut flashes the player's death sprite and then fades away within less than a second before resetting the level. This is enough time to show the player they died, but not too much time so they lose interest.

Super Meat Boy may be purchased to play at http://supermeatboy.com/ for various platforms.

Mr Jump

Figure 6: The title screen of Mr Jump for the Apple TV.

Mr Jump is a sidescrolling runner designed for the Apple TV. The avatar runs at a constant speed to the right while the player taps the controller to jump and avoid obstacles. Mr Jump has many positive reviews and has even been listed as Editors' Choice on the App Store. One reason Mr Jump has received such acclaim can be attributed to its minimalistic isometric graphic art style. While the game is played in two dimensions, the environment feels more rich with assets drawn to mimic the camera looking at the avatar from a slightly elevated angle. This 3D isometric art style mimics depth in an otherwise flat game.

Figure 7: Mr Jump features an isometric design where the avatar looks as if they are standing on a 3D platform, but they can only move within 2D space.

Micronaut's graphic art assets also follow a 3D isometric style with its platforms and various forms of scenery. However, the default physics engine has the avatar stand above the assets instead of on top of it (as if the camera was looking slightly from above). To remedy that, the collision bodies of the ground tiles have their tops cropped. Furthermore, scenery assets are placed offset similar to the avatar as to keep the isometric design consistent.

Mr Jump may be downloaded to play on the Apple TV or an iOS device for free at https://itunes.apple.com/us/app/mr-jump/id955157084?mt=8.

Super Mario Bros.

Figure 8: The title screen of Super Mario Bros for the Nintendo Entertainment System.

Super Mario Bros. is often acclaimed for the precedents it set in sidescrolling platformer video games. Most notably is how level 1-1 is designed to teach the player how to play. There is no need for dialogue boxes or instructions, but rather that player organically messes with the controller in a safe space and learns the controls naturally. Over time, the obstacles become more intense as the player becomes more of an expert. A Youtube video by Extra Credit delves deeper into the level design of the first level in Super Mario Bros. and also inspired parts of Micronaut's development.[3]

Micronaut adopts the same methodology of organically teaching players its controls by designing levels to introduce controls one at a time. While Micronaut uses text and images in the background scenery to show the player how to use the controller, the player is given a safe space to experiment with the new control. This leads Micronaut to be accessible to players of various levels and organically teaches the player to master the controls over time.

Super Mario Bros. may be purchased on the Nintendo Virtual Console on the Nintendo Wii or Wii U. More information about Super Mario Bros. on the Virtual Console may be found at http://www.nintendo.com/games/detail/3AhiHlPhEtLc5rGACE1dxueM0y5QDqCZ.

Features

Controls

Micronaut's control scheme is tabulated below.

Input	Action
Swipe Right	Run Right
Swipe Left	Run Left
Swipe Up	Jump
Swipe Down	Stop Running
Tap Touch Surface	Toggle Size
Home Button	Return to Apple TV Home Screen

Figure 9: Micronaut's tabulated control scheme. The left side shows what the user did and the right side shows what the result should be.

All of the controls are communicated to the player via animated billboard scenery one level at a time. The controls are introduced one at a time as to not overwhelm the player and allow them to organically learn the control by utilizing it within the level they learn it.

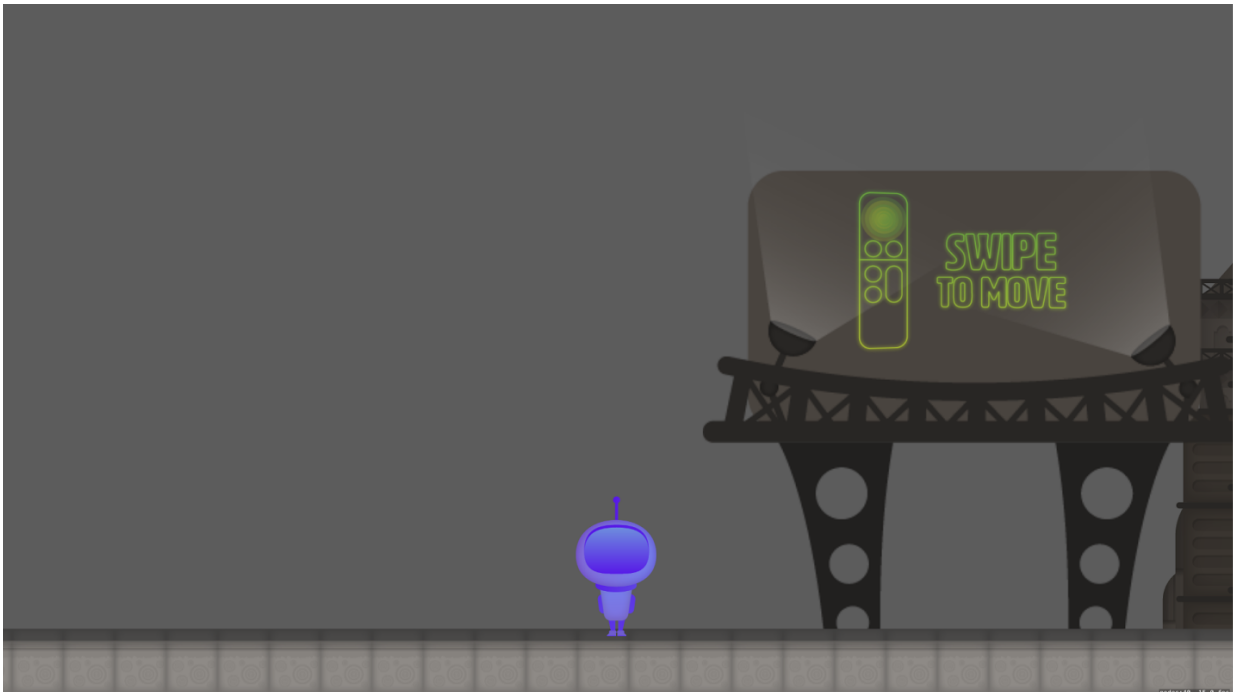


Figure 10: The first screen the player sees in Micronaut. The billboard animates highlighting the action of swiping the remote to move.

Micronaut's controls are designed to give the player more control over avatar and dimension of interaction. For instance, tapping the touch surface allows the player to change the avatar's size. Changing size gives Micronaut an extra dimension of interaction that is distinct from running and jumping. Multiple dimensions of interaction help Micronaut stand out from many other games on the Apple TV since most games only feature a single dimension of interaction. The most common example of a game on the Apple TV with a single level of interaction is one where the avatar runs at a constant speed and the player chooses when to jump to avoid obstacles. Micronaut aims to provide a richer user experience by utilizing a control scheme with multiple dimensions of interaction that work with each other to build a more vibrant game.

Gameplay

The gameplay is designed for a casual player who may be unfamiliar with video games. Therefore, each level is simplistic in its design and focuses on one mechanic at a time. This way the game comfortably builds upon mechanics taught within previous levels to ease the player into becoming an expert of Micronaut. Each level is described in detail below.

Level 1

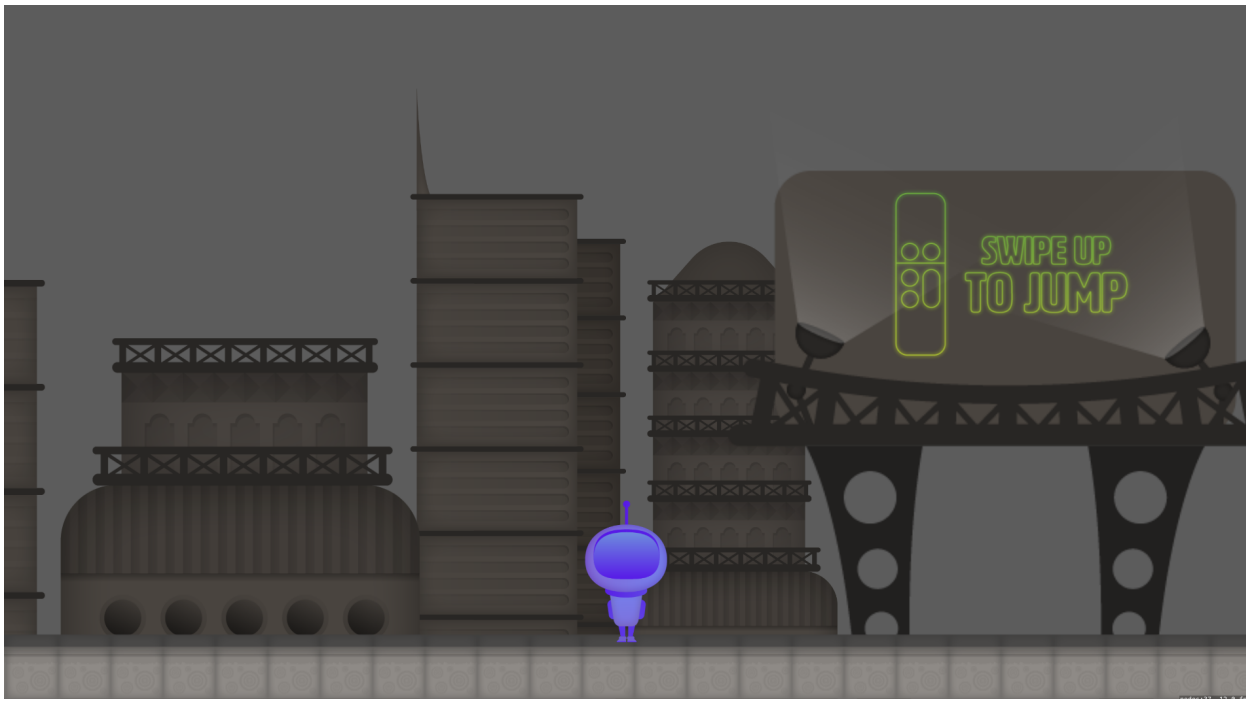


Figure 11: Billboards introduce basic controls to the player as they navigate the first level.

Level 1 is the first thing the player sees when they start the game. As such, it's layout is very basic with animated billboards in the background to introduce the two most basic controls: running and jumping. The level contains a wall and pit for the player to jump over before they continue to the next level, so they prove they understand the controls before advancing.

Level 2

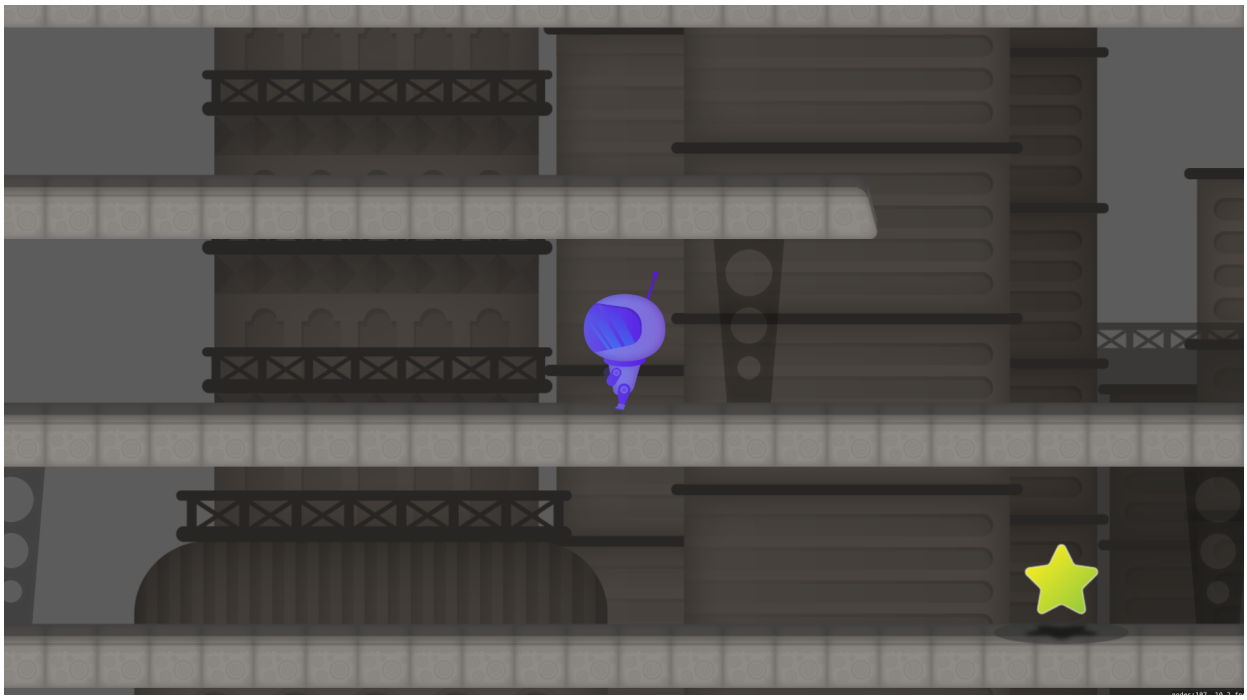


Figure 12: The player runs left to navigate the maze.

Level 2 shows the player they can swipe left to run in that direction. This is done with a vertical snaking pattern that requires the player to navigate right-left-right to reach the star.

Level 3

Figure 13: The player shrinks to navigate underneath the fire hazard.

Level 3 introduces the next main mechanic of the game: shrinking. The player is unable to advance past the first section of the game without shrinking to navigate underneath the hazard safely. Next, the player jumps over the wall and discovers they jump lower when they are small. Lastly, the player must tap to make themselves big again to jump over the spiked wall hazard at the end of the level before reaching the star. A second billboard instructing the player to change size is placed at the spiked wall hazard to show the player they can change their size back.

Level 4

Figure 14: The player waits for an opening while the enemies fly up and down.

Level 4 introduces the ability to stop moving. The level opens to a billboard showing the player they can swipe down to stop followed by a raised platform for the player to jump onto. To the right of the raised platform there is an enemy flying up and down in a regular rhythm occasionally blocking the player. This entices the player to stop moving until the enemy moves out of their way so the player can jump over it to advance. There is also a second enemy the moves a bit faster, so the player must have more precise timing to continue. The star is found after the second enemy.

Boss Level

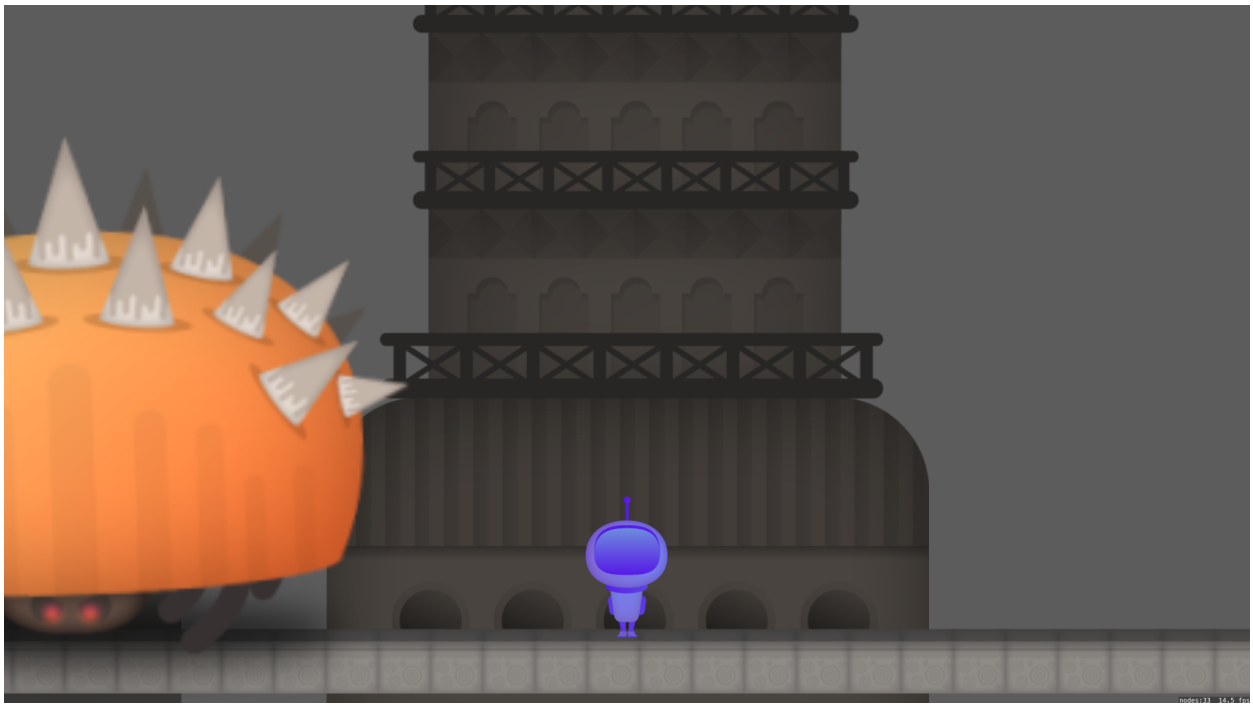


Figure 15: The boss waits for the player to move before it starts the chase.

The boss level is the final level of the game and it challenges the player to utilize all the skills they've learned. The level opens to a long empty stretch up in the air. After running past that stretch, the player drops down and watches a cinematic of the boss crawling forward then popping out of its shell. After the cinematic, the boss waits for the player to move before it starts the chase. The boss runs at the same speed as the player and if it touches the player then they are killed. To complete the level the player must overcome four obstacles, one right after the other, fast enough so the boss doesn't catch up. To keep the game engaging, the cutscene is skipped for future iterations of the stage. After completing the boss level, the animated banner "CONGRATS" appears on screen and the player is sent back to the beginning of the game.

Figure 16: After completing the game, the animated banner “CONGRATS” appears on screen and the player is sent back to the beginning of the game.

Respawn

To keep the game accessible to casual players, the player is not harshly punished for dying while trying to complete a level. The player may die by falling off a platform into a pit or colliding with a hazard. Furthermore, there is no timer which is typically seen within most platforming video games. A timer is often introduced to challenge the player to complete the level with a certain level of expertise before they are allowed to move on. If the player does die, they are returned to the beginning of that level. The player is given infinite lives so there is no sense of “game over” besides winning or restarting the game. This system is inspired by Super Meat Boy which has received critical acclaim for it’s fast paced respawn system which allows the player to challenge the level continuously without any harsh repercussions.

Figure 17: When the player dies in Micronaut the avatar is switched to the death sprite and fades away within a second before resetting the level.

Environment

The environment features an isometric 3D design inspired by Mr Jump on the Apple TV. This is accomplished in Micronaut by shifting the sprite collision body of all the ground objects down 10 pixels, so the avatar stands in the top portion of the ground image asset. Furthermore, the scenery assets are placed offset from the top of the ground so they look isometric as well.

Secondly, Micronaut features five layers of background parallax with varying degrees of scrolling speed. This gives the player the sense of looking out into a deep expansive background that reacts to their position on the screen. A still image of the background parallax within Micronaut can be seen in the image below.

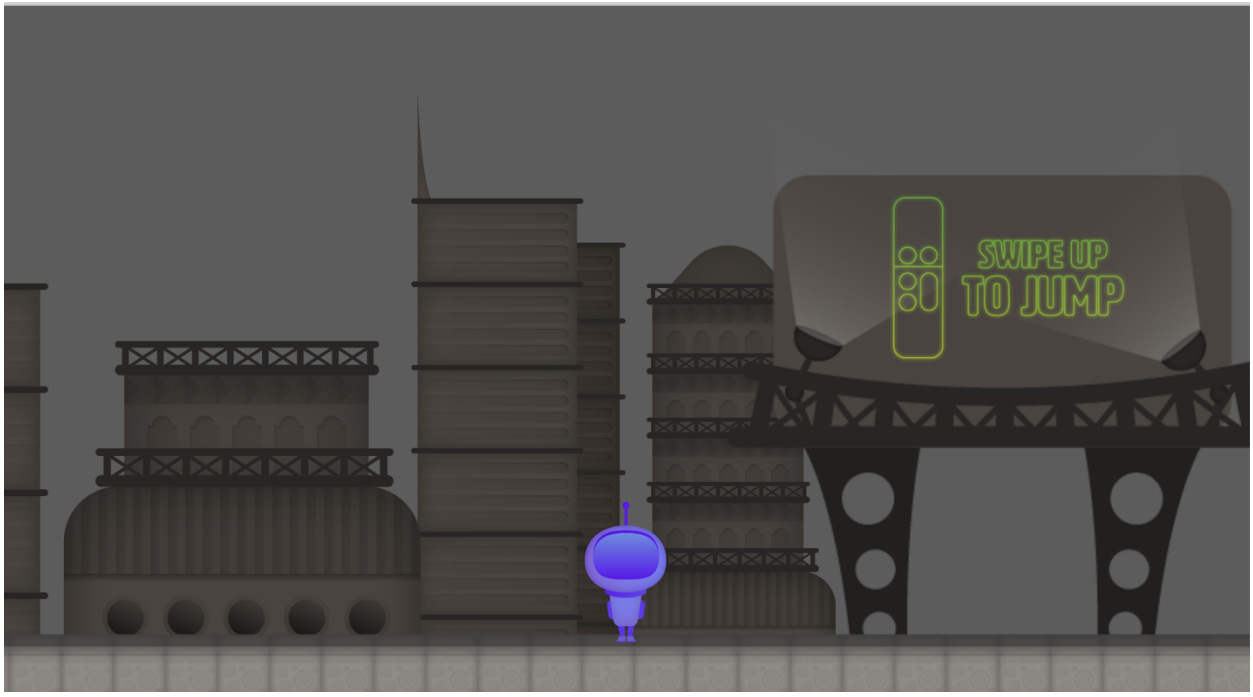


Figure 18: An instance of the background parallax within Micronaut. Each building is on a separate layer, so they scroll along at different rates.

Software Architecture

Micronaut's source code is structured with static classes, each of which fulfill a specific purpose. Below is a list of each of the important classes and their purpose.

- GameScene - handles the game loop and some events (e.g. two sprites collided)
- Utility - calculates the amount of time that has passed since the last update loop, overloads operators, and provides mathematical functions
- Constants - contains a table of constant values that are referenced throughout the game
- Controller - handles input from the remote
- Camera - handles positioning the camera
- World - keeps a map of nodes in the scene and updates them accordingly
- Player - handles the game logic in association with the player
- Boss - handles the game logic in association with the final boss
- Sound - handles playing and stopping soundtracks

Using static classes grants easier access to elements and methods throughout the game and ensures there is only one instance of each class within the memory. This design paradigm also removes the need for modules to be arranged in a hierarchical order so classes may reference each other cyclically. However, inheritance is limited with static classes since two classes may not inherit attributes from a static superclass without overriding the data from each other.

Xcode Scene Editor

When developing games with SpriteKit within Xcode, the developer may either programmatically or visually create and modify nodes with the scene editor (pictured below). Micronaut utilizes the visual approach for most of the nodes within the scene, except for important nodes such as the player, camera, and final boss that require a greater level of manipulation. Within the scene editor, nodes may have their physics body properties modified statically which is extremely useful for elements such as the ground or background scenery. Furthermore, nodes may be assigned static animations that may either repeat infinitely or occur at a certain time. However, animations may only be triggered through the scene editor based on game time and not on certain events. In order to reference important nodes, they are searched by name from the SKScene object with a double backslash so it ignores which level the node is on within the scene.[2] This way, nodes may be placed with the scene editor and possibly referenced programmatically for further manipulation.

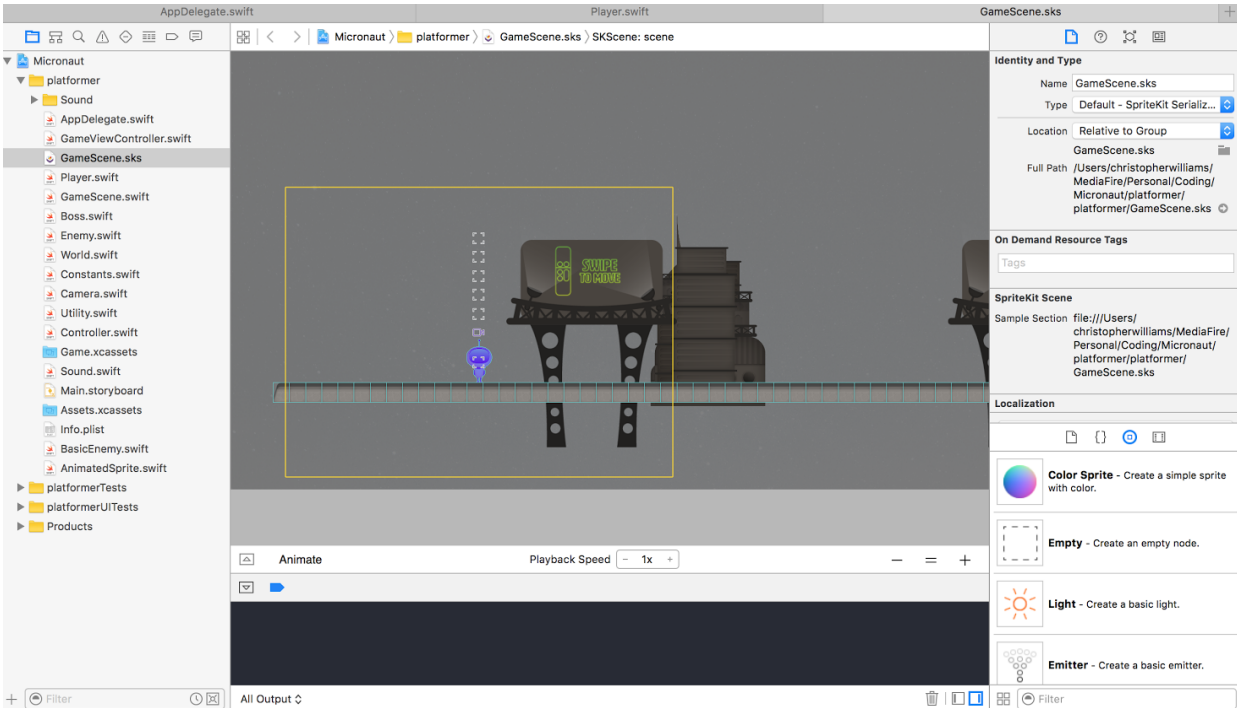


Figure 19: The Xcode scene editor allows developers to build a game visually by placing nodes in the scene and possibly even attaching animations to them.

Results

Overall, Micronaut was designed to give the player a great sense of control and engagement with the simplistic controller for the Apple TV. It explored the boundaries of human-computer interaction within the scope of a side scrolling platformer video game by assigning a distinct

action to tapping and swiping in each direction, but it kept the interactions feeling organic by not adding too many controls. Micronaut currently features five levels, each of which focuses on one mechanic to teach the player how to be an expert at playing Micronaut.

Reflections

Software Architecture

I started developing Micronaut with a disciplined sense of software architecture by keeping functionality modular and efficient. However, into the second quarter of this project I found such discipline for a single person developing a game hindered progress too much, so I began using the quickest and dirtiest methods to adjust the logic of the game to add more features. For instance, instead of having the final boss subclass the AnimatedSprite class, I copied a lot of the functionality because a bug arose where the Player and Boss classes were mangling each other's variables since they are both static.

This shift in discipline did allow me to add new features much more quickly (especially since I was the only developer), but the technical debt did start to compound near the end of the second quarter when I had to find tricky ways to add new features without ruining patched in features from the past. If I were to continue developing Micronaut, I would spend time paying back that technical debt by redesigning the logic to remain consistent.

Xcode Scene Editor

Early on in development, I made the decision to use Xcode's scene editor to layout sprites and design levels. I found alternatives such as Tiled which seemed to better suit my purpose of level design, but those tools were designed for Objective-C and iOS instead of Swift and tvOS. I could have used a JavaScript library to still use Tiled in Swift, but at that point I felt the hassle was not worth the reward. Besides, I wanted to see how Apple's native scene editor in Xcode would work for a 2D side scrolling platformer utilizing SpriteKit.

I found the scene editor to be very frustrating at times, but it does have its merits. I enjoyed how the scene editor allowed me to modify characteristics of nodes in the sidebar instead of programmatically searching for those nodes by name and changing their characteristics in the code. Furthermore, I am able to assign animations to nodes in the scene that are not dependent on a certain event such as the flying enemies in the fourth level. Lastly, I found the interaction with the scene editor to be very fluid and natural (when it wasn't lagging). Being able to drag and drop components and find assets and different types of nodes in the sidebar make modifications to the level design quick and easy to implement most of the time.

However, despite the merits of the scene editor I still found it frustrating. It takes up a lot of CPU and memory resources which lead it to lag a fair deal of the time. I used my fairly new Macbook Pro that I bought in September 2015 with a 2.9 GHz Intel i5 processor and 16 GB of RAM for developing Micronaut and whenever I opened the scene editor my machine would start to chug. While it wasn't unbearable, I find it a bit ridiculous that the scene editor caused a computer with modern specs that is less than a year old to slow down so much. Furthermore, whenever I accidently opened more than one instance of the scene editor (which is as simple as opening the SKS file within Xcode in a separate tab) it would more often than not toss all of my work in that file. When that would happen, I would have to revert to a previous commit on Github and redo that lost work over again. Lastly, I wish the scene editor had some functionality to quickly place down multiple sprite nodes in a tiled fashion to a specified precision. This is the main feature that made me debate using Tiled over the scene editor. After testing a level with stretched assets to see if it feels right, I would have to manually and precisely replace those stretched assets with tiled assets one by one.

I would personally recommend the scene editor for small levels and if you are looking to not tile your level design. Otherwise, you should look into using a third party tool such as Tiled.

Working With Jacob

At the beginning of the second quarter, I tried building a richer environment in Micronaut with better graphical assets instead of the placeholders I had in there at the time. However, despite my novice experience using Gimp, I found building graphical assets by hand too me too much time. To remedy the problem, I enlisted the help of a friend, Jacob Johannessen, to provide graphical assets. He was willing and very helpful not only as someone to provide assets for the game, but as someone to brainstorm with. Thanks to his help, we were able to discuss level design, mechanics, and the name of the game: Micronaut. Overall, I would highly recommend enlisting the help of someone to provide graphical assets not only to split the workload, but to have someone to provide a unique perspective when designing the game.

Overall, I wish I could have done more with Micronaut, but I am satisfied with the level it's at now and the skills I learned from developing it. It successfully shows a tvOS game can be more than an iOS clone by utilizing the controller in a way that would be intrusive on the user experience if Micronaut was on an iPhone or iPad. Furthermore, I got to teach myself how to develop for tvOS with Swift which can carry over to iOS development which has a very valuable market for developers in the industry right now.

Future Work

With any game, there is plenty of room for improvement and further development. First, Micronaut can use a title and pause screen to improve the pacing of the game and allow the player to take a break whenever they want without exiting the game. Second, an overworld map

would also create a better pace and allow the player to easily replay past levels and see their current progress in the game. Thirdly, there can be more levels, a deeper environment, and more story that is communicated to the player non-intrusively to have a more engaging experience. Lastly, SpriteKit's sprite collision framework for alpha masked physics bodies does not work well, so creating physics bodies manually for assets that are not well contained by a bounding circle or rectangle would make the collision more accurate. There are currently no plans to continue developing Micronaut at this time.

References

[1] "App Store Metrics." *Mobile Games Industry News, Discussion, Analysis, Opinion, Events, Jobs, and More*. N.p., Sept. 2012. Web. 25 May 2016. <http://www.pocketgamer.biz/metrics/app-store/>.

[2] "SKNode." *Class Reference*. N.p., n.d. Web. 25 May 2016. <https://developer.apple.com/library/ios/documentation/SpriteKit/Reference/SKNode_Ref/index.html#//apple_ref/occ/cl/SKNode>.

[3] Extra Credits. Design Club - Super Mario Bros: Level 1-1 - How Super Mario Mastered Level Design. *YouTube*. YouTube, 05 June 2014. Web. 01 June 2016. <https://www.youtube.com/watch?v=ZH2wGpEZVgE>.

[4] Faarkrog, Morten. "Introduction to the Sprite Kit Scene Editor." *Ray Wenderlich*. N.p., 2 Nov. 2015. Web. 25 May 2016. <https://www.raywenderlich.com/118225/introduction-sprite-kit-scene-editor>.

[5] Wenderlich, Ray. "Sprite Kit Swift 2 Tutorial for Beginners." *Ray Wenderlich*. N.p., 30 Oct. 2015. Web. 25 May 2016. <https://www.raywenderlich.com/119815/sprite-kit-swift-2-tutorial-for-beginners>.

[6] "SpriteKit Programming Guide." *Advanced Scene Processing*. N.p., 31 Mar. 2016. Web. 25 May 2016. <https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Actions/Actions.html>.

[7] "Looping Music | Soundimage.org." *Soundimageorg*. N.p., n.d. Web. 25 May 2016. <http://soundimage.org/looping-music/>.

[8] "Freesound.org - Freesound.org." *Freesound.org - Freesound.org*. N.p., n.d. Web. 25 May 2016. <https://freesound.org/>.

Appendix

Installation

An Apple TV (4th generation or later) or an Apple developer account and Xcode with the tvOS SDK is required. Either clone the Git Repository and run the game through the Xcode simulator or find the game on the App Store after it is approved by searching for “Micronaut” on your Apple TV.

Git Repository

The source code may be found at https://github.com/RetroRebirth/Micronaut.

Feedback Survey

The following Google form was utilized to gain feedback on various aspects of Micronaut: http://goo.gl/forms/38yOqhWDdNHbdMxp2. Below is the tabulated results from the survey from six participants.

Category	Average Score (1 to 5)
Difficulty (3 is Perfect)	3.0
Enjoyment	4.3
Performance	3.8
Controls	3.0
Art	4.3
Music	4.5
Level Design	4.0
Game Mechanics	3.5

Figure A-1: The tabulated averaged results from the feedback survey.

In the textual responses, most of the participants were frustrated with the Apple TV remote since it has a tendency to not recognize the swipe up input. Secondly, participants requested there were more levels to the game. Overall, participants said they found the game just challenging enough to keep them interested and wanting to complete it.

Prototype Designs / Illustrations

Figure A-2: One of the first sketches for Micronaut. The idea was to have the player explore a barren landscape and visit a catina with interesting characters similar to a RPG.

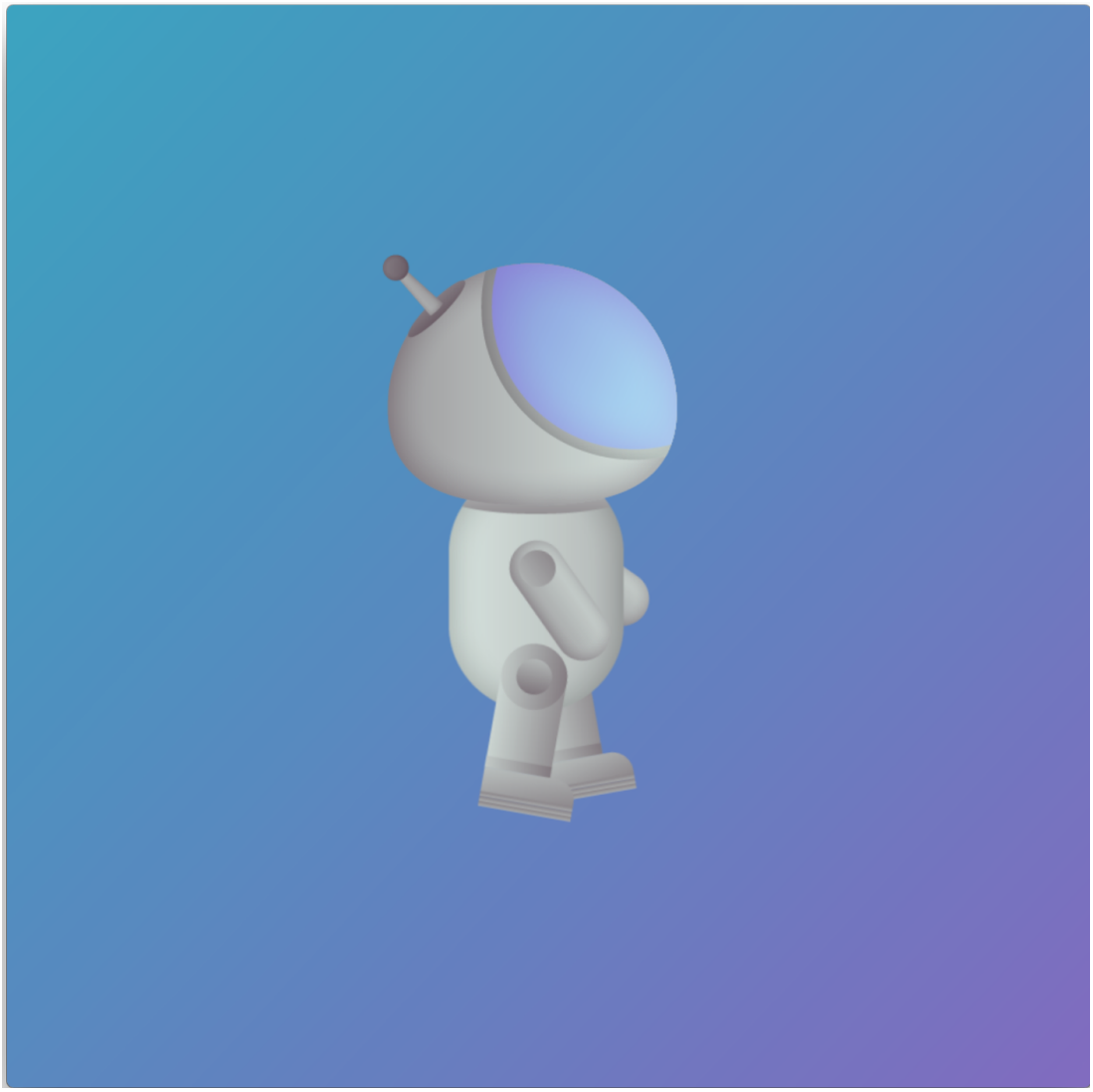


Figure A-3: The first design for the player avatar, the appendages were originally planned to be separate sprites that would be layered and manipulated within SpriteKit.

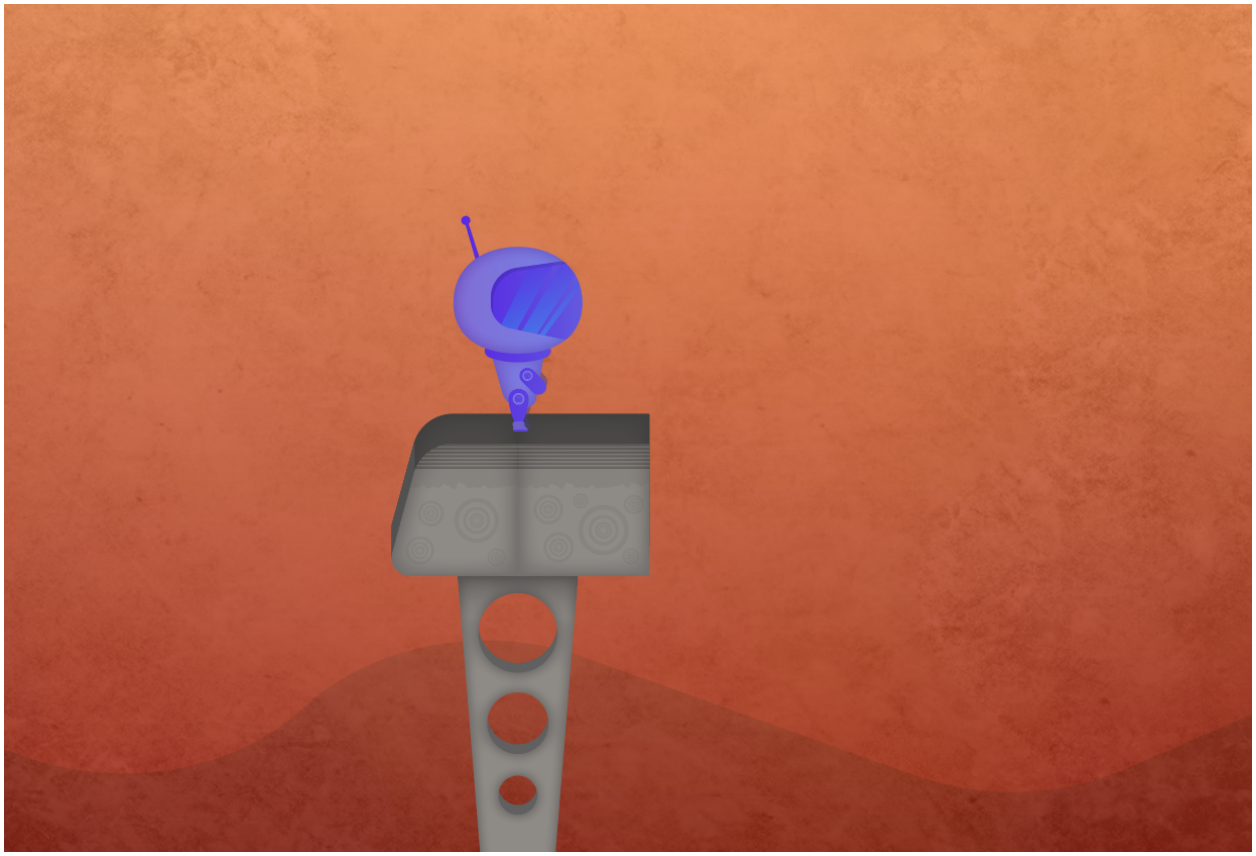


Figure A-4: A basic arrangement of the assets find in an early version of the game. These assets provided the framework to experiment with level design.