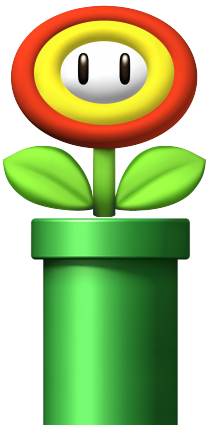
SUPER MARIO RUN

Report Leader: Bikram Chatterjee

Team Members: Hung Le, Jiaxu Tian, Dallas Reynolds



INDIVIDUAL CONTRIBUTIONS

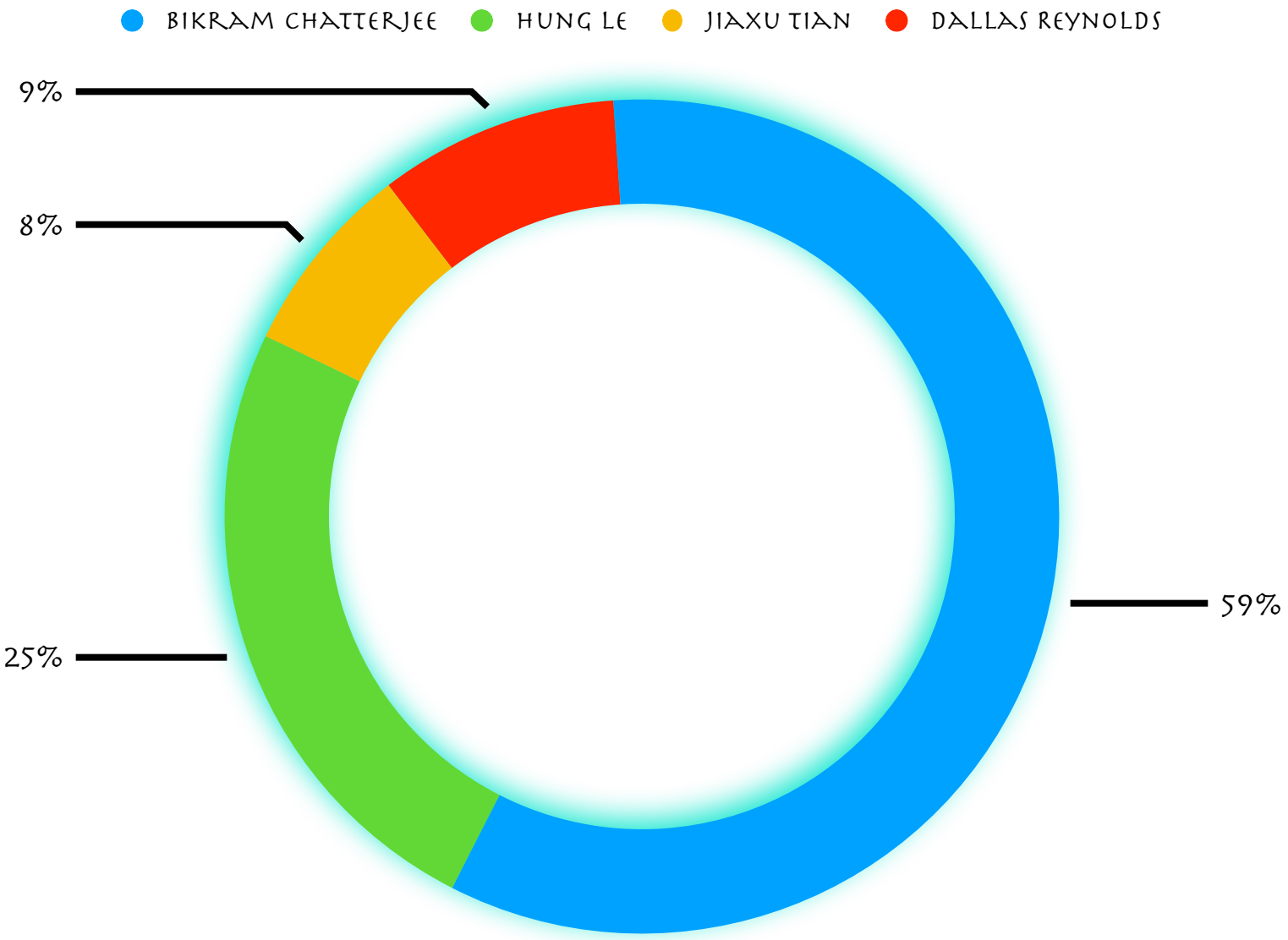


TABLE OF CONTENTS

SUMMARY OF CHANGES

Project Objectives

- Removed Wall Jumping feature
- Removed all enemies except Goomba
- Added Developer Mode
- Added jumping in instructions
- Added Pause Menu, Game Over Screen and Level Complete Screen
- Changed Luigi to Yoshi

UML Diagram

Additions:

- ScanLevel: to read the text file
- Level: takes the hashmap with the coordinates of the objects and places it into the environment arrayList
- endPane: Option the user has to view highScore and continue onto the tour Pane
- losePane: A screen that pops up when the user dies. They can retry or go back to the tour pane.
- lvlPaneDev: An option that can be used to view coordinates and scroll across a level without having to play it.
- pausePane: PausePane gives the user the option to pause the game and either retry, go back to the tour, or continue the game
- Character Superclass
- eGoomba

Deletions:

- audioPlayer
- graphicsPane
- graphicsApplication
- mouseEvents
- soundEffects Package
- PowerUp Select is used to in the shop, does not have separate option.
- InventoryPane

- LevelInterface
- levelMaker Package
- power-up Interface
- enemies Interface
- Character Interface

Use Case Diagram

- Changed to proper bubbles and actor stick figure
- Deletion of power up and character choice before each level

Sequence Diagram:

- Character Sequence Diagram
- Star Power-Up Sequence Diagram

Algorithms

- Removed wall-jumping feature
- Added mouseCounter to check how long the moues was held for
- Added jumpCount in order to see how high the user has jumped
- Added jumpLimit in order to set a maximum height the user can jump
- Added fallDown() in order to check the collision with the floor and allow the character to come back down
- Removed height++ and character move speed since the background is moving not the character.

CUSTOMER STATEMENT OF REQUIREMENTS

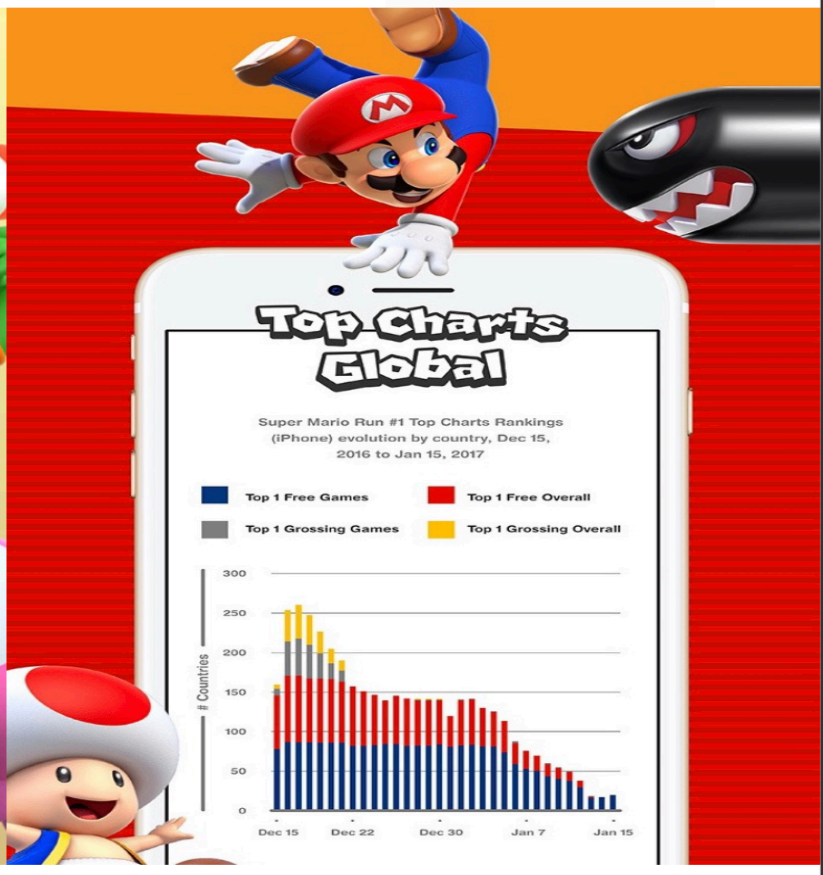
(A) ABOUT SUPER MARIO RUN

Super Mario Run is the perfect game for people who want to escape the struggles of the real world for a desired time and explore the fun and lively world of the Mario-verse. As someone who avidly plays video games, I know that Mario Run is easy to get the hang of. For someone who plays video games often, or someone who rarely plays. The mechanics of the game are smooth because of the physics engine that we have personally designed. As more users play Mario Run the physics of the game make it apparent that the game runs nicely.

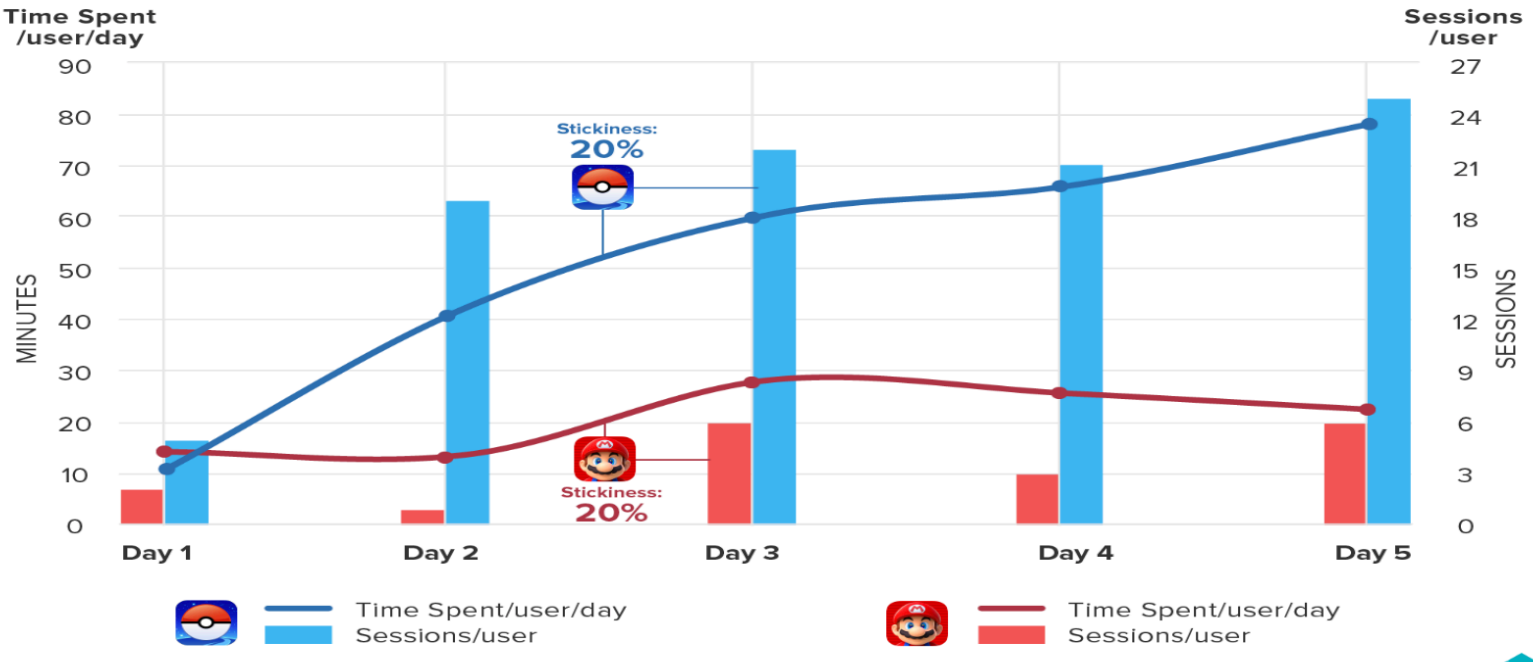
In the gaming world, there is a hole for a good and updated Mario game. We started off by creating the graphics for the game which then allowed us to add the basic functions for the game. This allowed us to build the movement of the character and enemies around the level. Therefore shortening the process and making it more cost-effective. Step two was where the menus, loading screens are formed and developed. This is a very important part of our game. These menus and their functionality will be the determining factor of whether people remain interested in the game. The menus provide external enjoyment in the game by allowing the user to buy skins and power-ups. The third step was the user interface and the player movement. Completing this early on was important to us in order to make the levels playable for the user. However this took the longest. We wanted the movement to work well and it took most of our time to implement.

“Super Mario Run is not the first iOS game that was released on the App Store and became an instant hit, its brand is well known and goes back to mid 80s. So it’s really interesting to compare its first day revenue to the revenue of other recent mega hits – Pokémon GO and Clash Royale. Within the first day Super Mario Run generated almost twice as much as Pokémon GO, $5 million against $3 million and five times as much as Clash Royale, which generated only $973k” (Business of Apps, 2017.). Mario Run was a very successful game. However, it died off quickly unlike its competition Pokemon Go and Clash Royale. Pokemon Go has many features that go beyond what Super Mario Run offered. Like the ability to collect many different Pokémon and to upgrade them. Clash Royale has many characters to unlock and to experiment. These are key features that our version of Mario run will have to ensure a long-lasting game that has months of replay-ability. We give the user the ability to purchase different characters to play as. There are also two power ups to choose from which have different effects. Which makes our Mario Run a great investment opportunity. “Because the game was released globally, in 150 countries, it’s interesting to analyze in how many countries it lead the Free Games, Free Overall, Grossing Games and Grossing Overall charts. On the day of its release, December 15th, for half of 150 countries it was number 1 on Free Games chart and the other half on the Free Overall. By January 15th Super Mario Run went down the charts significantly, being only number 1 in less than 25 countries and only on the Free Games chart” (buisnessapps.com). This data shows that with work this can be an even more profitable and a longer lasting game. With the current interests of people being games that are fast-paced and provide customization options; it is our goal to make Mario Run into something that has both action and immersion options available to the user.

(B) Supporting Statistics



Super Mario Run vs. Pokémon GO: the First 5 Days



Source: Verto App Watch™, U.S. adults ages 18+, 2016

verto analytics

GLOSSARY

- BGM: It is the abbreviation of "background music”.
- Coins: Coins (sometimes known as Gold Coins or Yellow Coins) are the main currency of the Mushroom Kingdom. Players can collect these during the game to unlock other characters and special power-ups.
- Controlled Jump: A jump where the player has to get mid-height.
- CP: Shortened version of "Checkpoint".
- GIF: It is the abbreviation of "Graphics Interchange Format", and it is a format allows graphic movements through a sequence of images
- Goomba: Goombas resemble small, brown mushrooms and are a fungus-based species. Goombas are physically weak and are not much of a threat to the player since a single stomp usually defeats them.
- Long Jump: A jump which allows the player to reach greater heights while jumping.
- Power-Ups: Items that provide extra abilities or cause appearance changes for Mario and other characters.
- Mario-Verse: An informal term used by fans to describe the entirety of the Mario franchise, often from an in-universe perspective.
- RPG: It is the abbreviation of "role-playing game", and that is a kind of game that player act a role in the video game.
- Super Mushroom: The super mushroom changes the size of the player’s character.
- Super Star: The super star gives the player 10 seconds of invincibility.
- Unlockable: An item that can be accessed after completing a certain task in the game.

FUNCTIONAL REQUIREMENTS SPECIFICATION

STAKEHOLDERS

- Developers : The developers are the four of us: Bikram, Hung, Dallas, and Jiaxu.

- Tester: In order to build a complete and well functional program, testers are needed for feedback. This will ultimately help us improve our program

- Users: The targeted users of this program are the people who have an interest in video games, specifically in the RPG or Arcade Genre. These players may fall into the following categories: mobile gamers, vintage game enthusiasts, Nintendo players and casual gamers.

- Manager: The manager of this project is our instructor Osvaldo Jimenez. He has assigned instructions and requirements for this program, and therefore he will also be holding us accountable for any failure to meet the specified requirements.

- Sponsor: The sponsor of our project is the University of the Pacific, because the school provides us scholarships. That helps us continue our study at the school and finish the program.

ACTORS AND GOALS

The person who will directly interact with the system will be the player, and this program is made for them. Our main goal for this program is to allow people to have fun. Mario is an interesting and classic game, and the time with this game is always full of fun and joy. We want to share this experience with more people because many people are stressed out in their everyday lives. Although the world in the game is virtual, the fun that people have is real.

USE CASES

Casual Description

1. Start Screen: After initializing the program, the user sees a screen that welcomes them to the game. The user clicks anywhere on the screen to continue to then main menu.

2. Main Menu: Once the user gets to the main Menu, they will have three options to choose from. They are able to click on Tour to go to the level select, Shop to buy characters and power-ups, and guide in order to view the instructions.

3. Developer mode: Once the user goes to Tour, they are able to click on developer mode. In developer mode, the user is able to click anywhere on the screen in order to see where a certain coordinate is. They are able to move the objects in the level around in order to see how the object would look at a different spot in the level. Finally they can click or hold the left or right button to move to different places in the level.

Formal Description

Star Power Up

1. User equips Star in the shop
2. User goes back to the main menu and clicks on Tour
3. User chooses any level they want
4. Once the level of their choice starts, the user will notice that the character has a star animation and the star music is played
5. The User will have 10 seconds until the star runs out.
6. If the user runs into an enemy with the star, they will not die
7. If the user runs into a wall and does not move, the star will still run out after 10 seconds
8. If the user jumps into a gap with the star, the character will die and the user will get sent to the losing screen.
9. Once the ten seconds are up, the star animation for the character will go away, the music will stop, and the user will go back to the normal character that they selected.

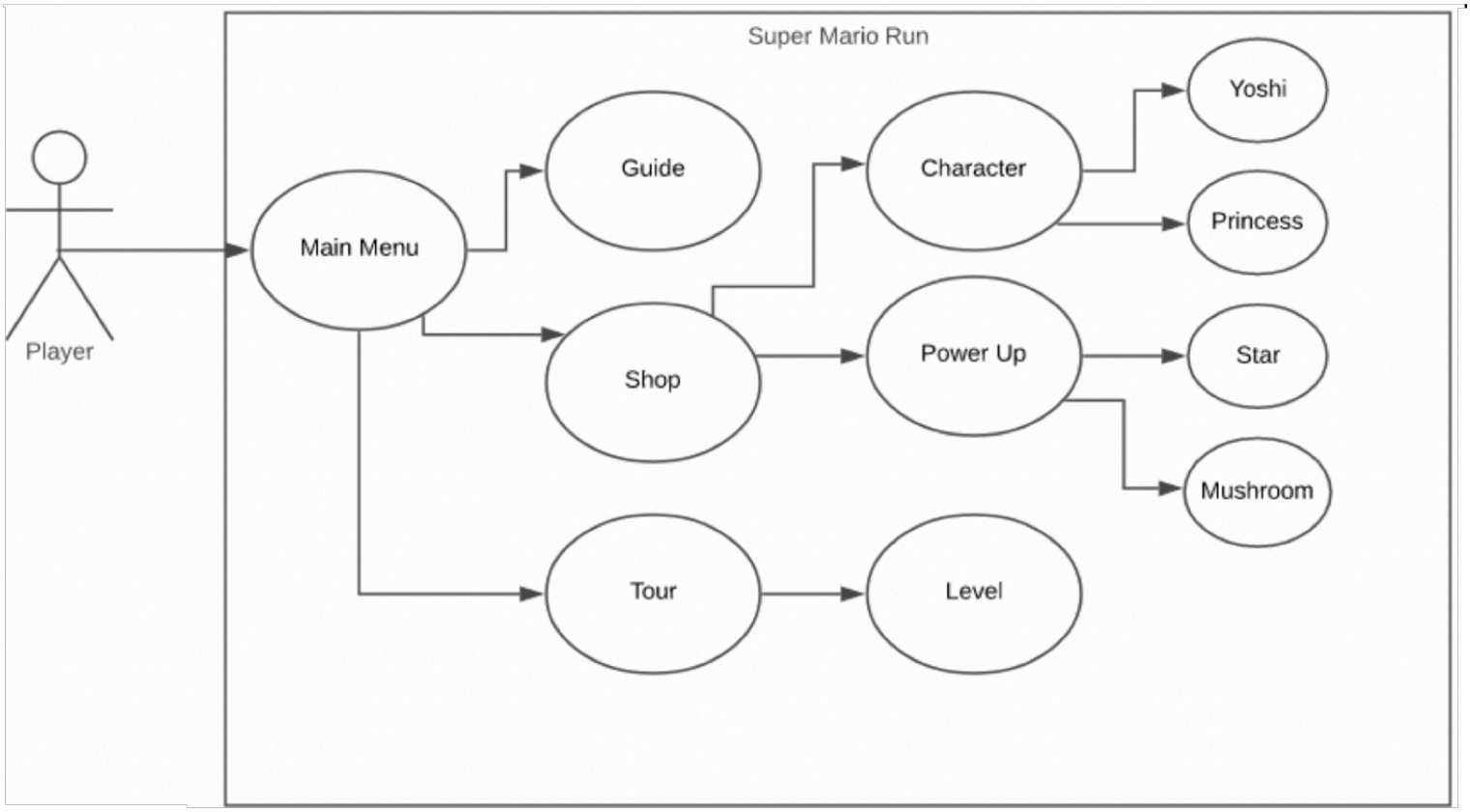
Shop

1. The user clicks on the shop in the main menu: The screen will transition from the main menu to the shop.
2. The shop will display “Characters” and “Power-Ups.
3. User clicks on “character”
    1. Once the user clicks on character, three buttons will appear. Two of those being buy for Yoshi and Princess, while Mario is “equipped button”. The total number of coins the user has will also appear.
    2. If the user clicks on “buy” for Princess or Mario, the button will change to “equip”
    3. If the user clicks equip for either Princess or Yoshi, the equipped Button will appear for the one that the user chooses to use while changing the the previous equipped character to equip.
    4. If the user wants to buy a character, but they don’t have enough money, the button will not change from “buy” to “equip.”
4. User clicks on “Power Up”
    1. Once the user clicks on power ups, star will appear with corresponding buttons to buy the power up. The total number of coins the user has will also appear as well as how much each power up cost.
    2. If the user clicks “buy” for either mushroom or Star and they have enough coins, the button will change to “equip”.
    3. If the user clicks on equip, the equip button will change to equipped and if a power up is already equipped that button will change back to equip.

Jumping

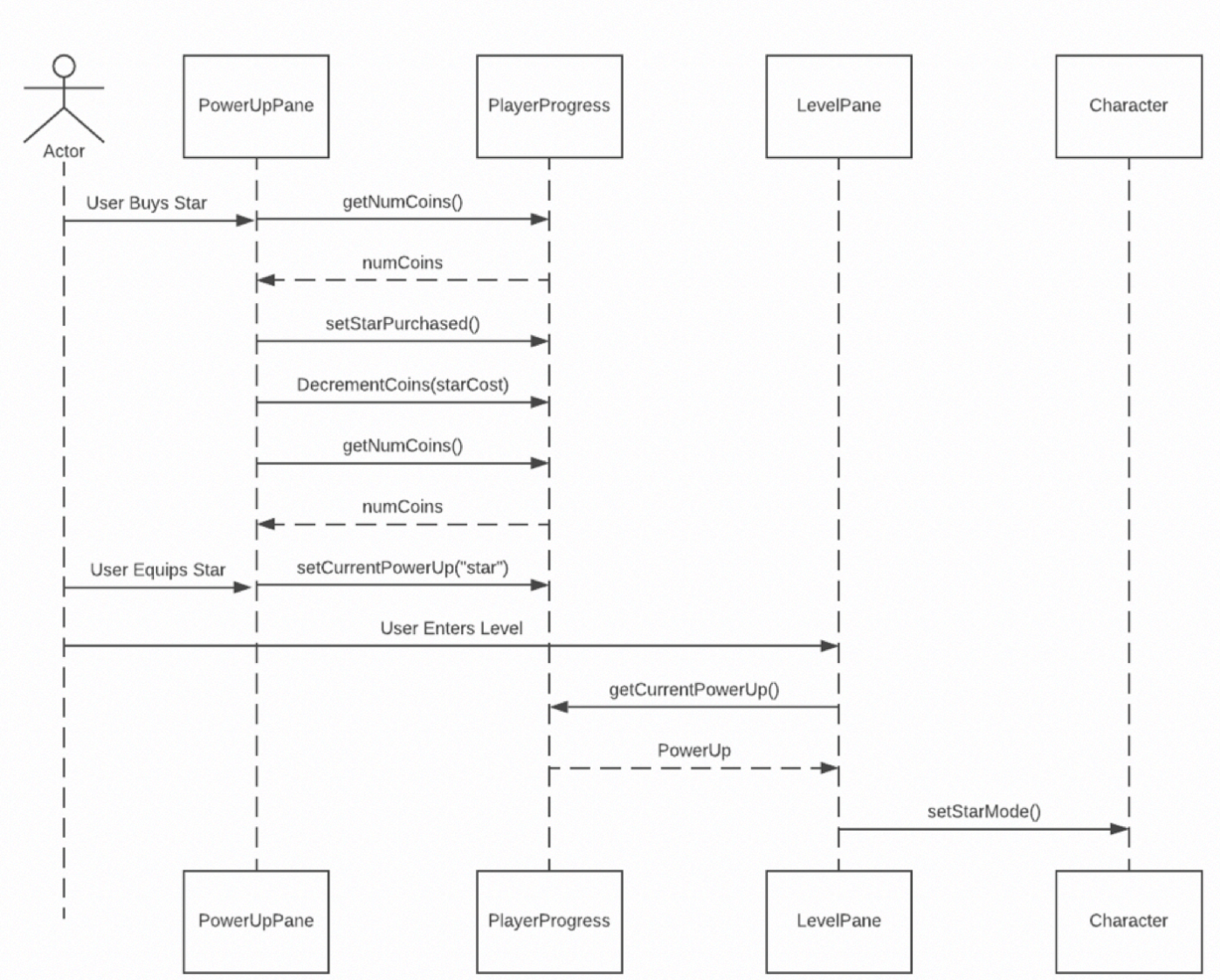
1. If the user lightly taps the mouse to jump, the character will do a quick short jump
2. If the user holds the mouse click to jump, the character will do a longer and higher jump
3. If the user runs into a wall without clicking the mouse, the screen will stay still until the user clicks to jump over the wall.
4. If the user jumps up and there is a platform on top of them, the jump will be cut short and the character will start falling due to the collision.
5. If the user Jumps on top of an goomba, the goomba will disappear.
    1. If the user jumps over the goomba, the goomba will walk pass the character and nothing will happen as the character will continue on with the level
    2. If the character jumps and is not directly on top of the goomba, the user will lose the game and be sent to the losing screen.
        1. If the character jumps and falls into the gap, they will die and be sent to the losing screen

USE CASE DIAGRAM

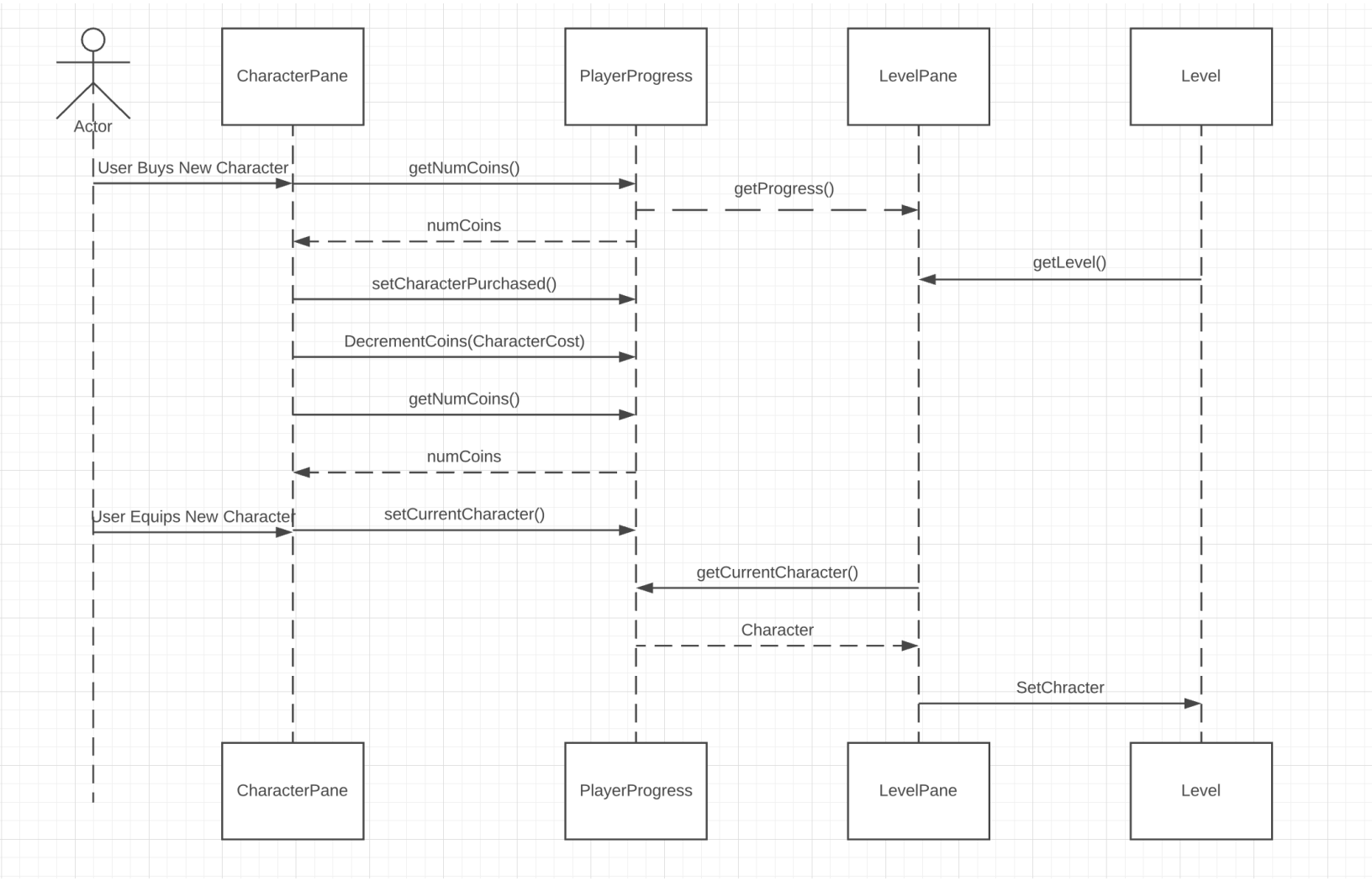


INTERACTION DIAGRAMS

(A) STAR POWER UP SEQUENCE DIAGRAM



(B) CHARACTER SEQUENCE DIAGRAM



UML CLASS DIAGRAM

(A) IMPORTANT CONTRACTS

1) getNumCoins(): Used to get current coins from playerProgress during pause menu, level cleared and the shop. This is important as it holds how much coins the user has and it keeps track of this number. This number can be used to purchase power ups and characters in the shop as well as to keep the high score of each level.

2) getFallState(): LevelPane calls this function from Character class to check if the character is still falling down

3) getCharacter(): Get character is used in order to set the character the user buys in the shop. Player Progress is used so that the character that the user buys is saved and it can be equipped and used in the level. Without this method in playerProgress, the user would not be able to switch characters or go back to other characters as they wish in the shop

4) playerProgress: The player progress is the second most important class as it is connected to almost half the the other pane classes which require data from playerProgress. For example levelPane needs info about which levels are unlocked and powerUps pane need to update playerProgress if new items are bought.

5) AudioPlayer: Our audio player class allows sound effects to be played throughout the game. It connects to all the panes, allowing the us to play music in each pane and level. The audio player is also used in each level for the sound effects. When the user kills and enemy, collects coins, or jumps, a sound affect from the audio player class will play.

6) Level: We will be using a single interface Level which connects to four other level classes. Each level class sets the location of each object. It is connected to levelMaker and lvlObjects to determine the location

(B) CLASS DIAGRAM

SYSTEM ARCHITECTURE AND SYSTEM DESIGN

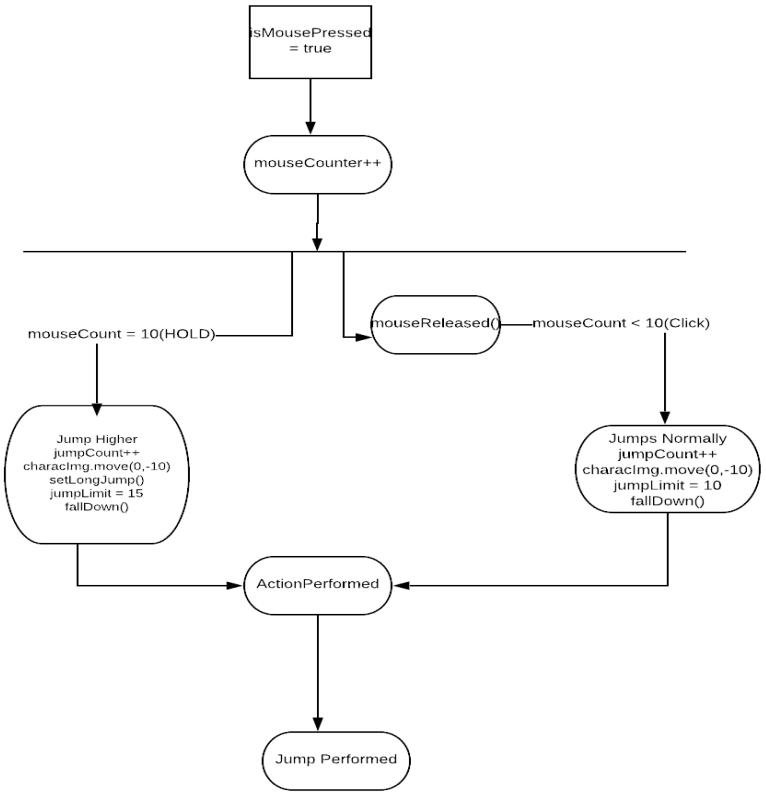
A) Mapping Subsystems to Hardware: Our system does not depend on a web browser or a web server. Our game is independent from the internet.

B) Network Protocol: Our system runs on a single machine so this does not apply to our game.

C) Minimum Hardware Requirements:
CPU: Mediatek MT6582M Quad Core 1.6 GHz or equivalent
GPU: ARM Mali-400 or equivalent
RAM: 1 GB
Storage: 500 MB
Display: LCD Color Display
Resolution: 1155 x 650
OS: Windows 10 version 1507 or higher/ macOS 10.14: Mojave (Liberty) or higher

ALGORITHMS

Precondition: mousePressed() is triggered in LevelPane



LONG JUMP ALGORITHM

To implement our long jump, we needed to calculate the amount of time the user holds the click of the mouse and trigger a long jump when the user has held the click for 300 milliseconds. By default, Java does not include any methods that allows developers to constantly detect if the user is currently still holding the click of the mouse after the mousePressed event is triggered and calculate how long it has been since mousePressed was triggered. However, Java does provide a method called mouseReleased to detect when the user has released the click of the mouse. To solve this problem we used our existing timer in the LevelPane class and introduced two variables isMousePressed (boolean, initialized to false) and mouseCount (int, initialized to 0). When mousePressed event is triggered, isMousePressed becomes true and the character class is called to trigger a jump. Now in actionPerformed, if isMousePressed is true, mouseCounter keeps incrementing every actionPerformed(). Now, if the user releases the mouse within 30 milliseconds of pressing it (i.e. mouseCount < 10), mouseCount gets reset to 0 and a normal jump is implemented. Otherwise, levelPane modifies the Character class's jumpLimit from 10 to 15 to implement a higher jump while the character is still in its jump state.

DATA STRUCTURES

We have used two data structures: HashMaps and ArrayList.

HashMap:

HashMap is a collection object which uses key and value pairs, where both parameters are objects declared on creation. Each key maps to a corresponding value. We are using the HashMap because it will make our coding process easier and faster. For example, we will be using a HashMap in order to store all the different environment objects and place it into a level. We are able to achieve this by using a complex HashMap which consists of a string (level object name) as the key and an ArrayList of Pairs of two integers representing the x and y coordinates of the level objects. The Scanner class will then read the file line by line and keeps adding object coordinates into the HashMap. Once the all the coordinates for all the objects are placed into the hash map, Level class to extract and sort each object and place it into the environment array where it will be displayed in the corresponding level.

HashMap has superior performance to TreeMaps and HashTable. HashMap has a performance of O(1). Where as HashTable runs at O(log(n)) which is slower. HashMap is more flexible because it allows for one null key and any number of null values. It also can utilize unsynchronized objects, which typically perform better than synchronized objects. In the case of our system a HashTable is very useful for it's superior performance and flexibility.

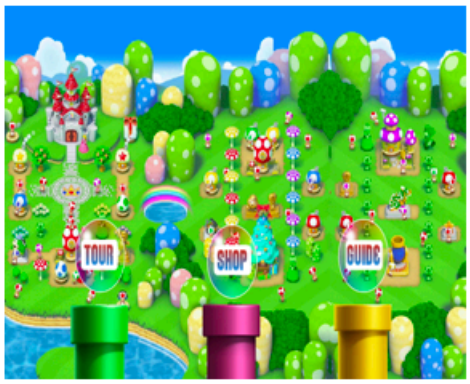
ArrayList:

ArrayList uses an array as its underlying implementation, so in effect you have the advantage of faster access to elements, while at the same time you are able to dynamically add and remove elements. One more advantage is that it is more flexible changing the underlying implementation of an ArrayList, for eg, if you want to make it synchronized, you can convert to a Vector without having to rewrite all the code for an Array. We will be using ArrayLists in order to store all the coins, platforms, walls and enemies for a certain level. We will use the ArrayLists to make the level maker. Using the level maker, we would design the placement of all the objects in the certain level we are designing. This would make it much easier when the user wants to restart a level from the beginning, and that is the reason we use this data structure. We can just call that certain array for that level in order to restart the game and return all the objects into its original position. Array Lists allow for direct reference to every element in the list. Array lists are faster than linked list which will make our program run more efficiently.

USER INTERFACE DESIGN

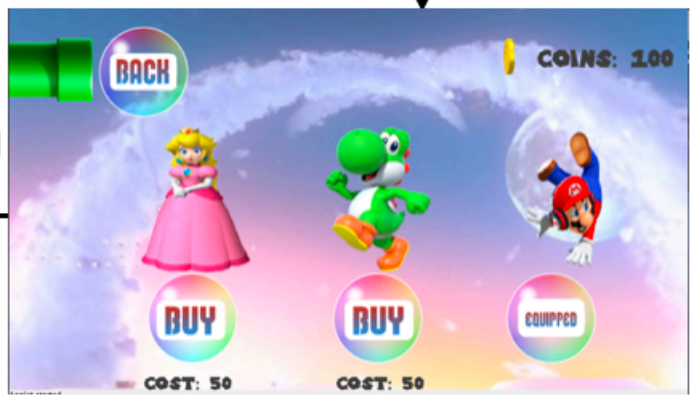
Updated Design

User Clicks once To Start Game

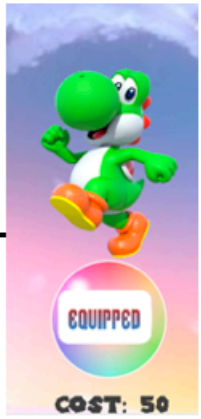


User Clicks Shop

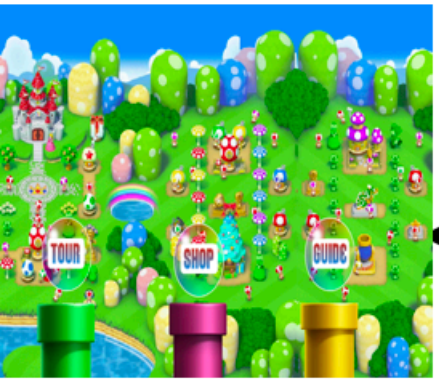
User Clicks Character



User Buys and Equips Yoshi

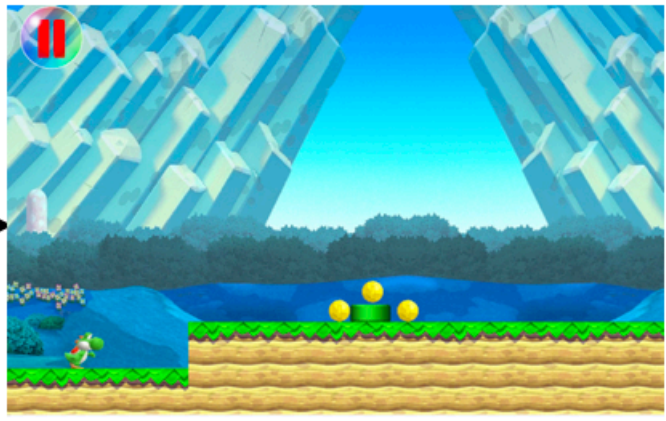


User Goes Back To Menu



User Goes To Tour

User Clicks The First Level



Updated Design

User Interface Design Differences and Similarities

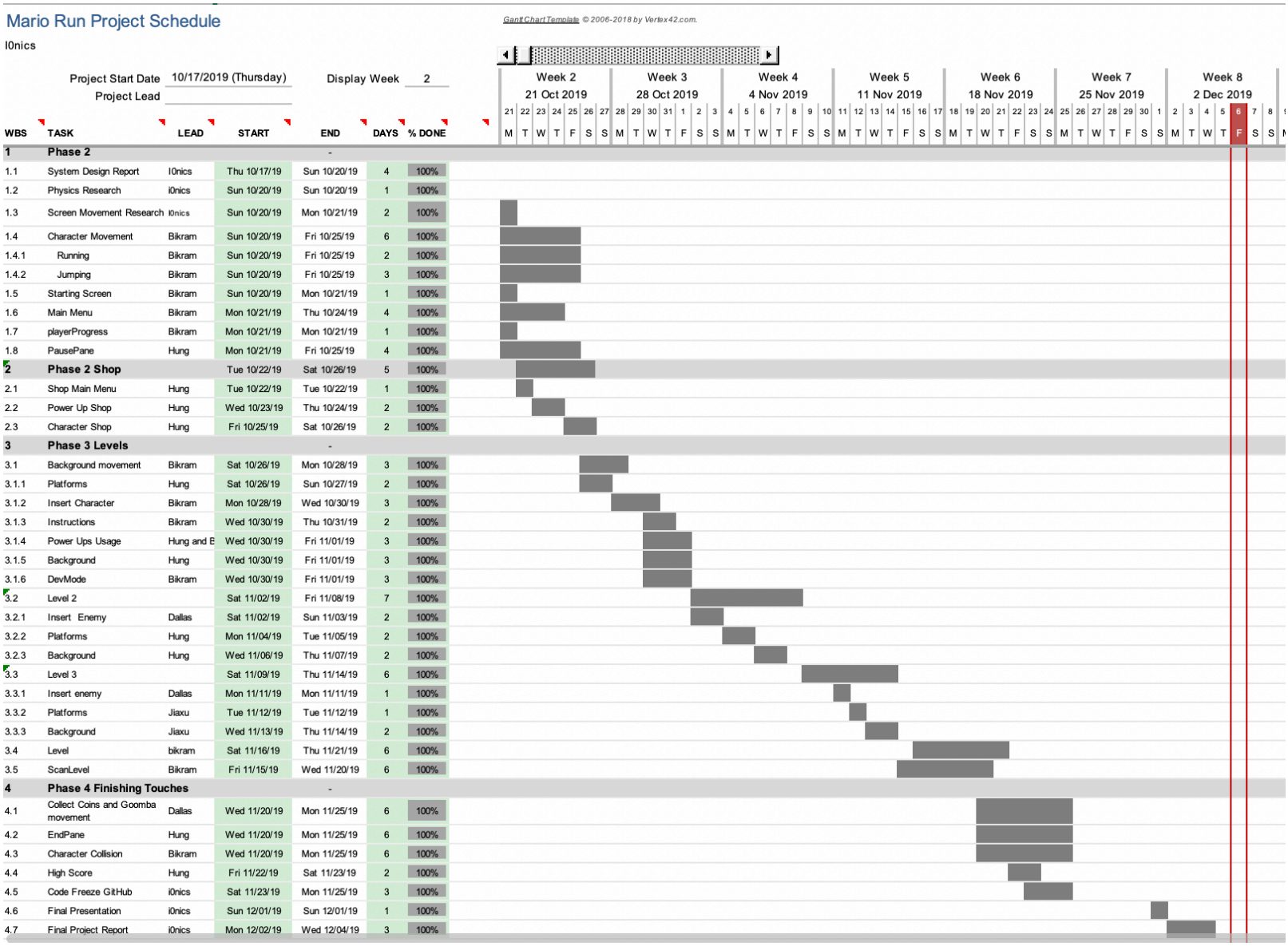
In the original diagram the user clicked tour, selected a level, and began playing. After this the user goes to the shop selects the power-up menu and buys a mushroom. The user then goes back to the main menu and exits the game.This is much different compared to the updated version. When the user starts the game they will see the start menu and they will click once to go to the main menu. When they are in the start screen the background has changed but the functionality of the start screen has not changed. If the user clicks the screen they will be taken to the main menu. Once in the main menu, the buttons and pipes have been altered and the background is now smoother and fits the screen better. Should the user click on the shop button they will be taken to the shop pane. In the first user interface diagram the next step was to go to the tour pane. It is now the shop pane first. In the shop pane there is an option for either the character screen or the power-up screen. The user clicks character and is taken to the character pane. Within this menu the user would then buy Yoshi and equip Yoshi. Then the user goes exits the game and goes back to the main menu. The user goes to the tour and clicks the first level. These are the changes that were made for the updated user interface design.

User Ease of Use and Effort Estimation

The user interface design is a key focus of ours. We want the game to feel easy to use and to navigate with ease through the menus of Super Mario Run. This is why all of our menus are very simple and the user does not need to do a lot of leg work to figure out how to play. Ever since we began designing the game, we created a detailed path of what the user will see and what it takes to get there. An example would be if the user wants to see what they can buy, they would click to start the game, which would send them to the main menu. Once the user is at the main menu, they have the option of clicking on the shop button which would send them to the shop. Once the user has gotten to the shop menu, two options will appear, characters or power-ups. The user then decides they want a power-up instead. So they would go to that menu and buy one. Then they could go back to the main menu and play level one. Originally, we were going to have a pop-up menu appear at the beginning of each level so the user could quickly equip the power-up that they have chosen. This confused some users during the paper prototype since they didn't know if they were supposed to click on the power-up. To negate this confusion, we decided to have the user equip the power up and allow them to have activated as soon as the level begins. This makes it easier for the user so they don't have to do this go into an inventory screen right as the level begins. If the user were to go straight from the start menu to the tour screen they could play level one and then have more coins to buy a character or power-ups. The amount of effort it takes is very minimal. The user only has to click 9 times in order to start the game, buy a power-up, and start a level. If the user wants to go straight into a level it would only take 3 clicks to get right into the game. For an average run-through of level one it takes 20 clicks to finish the level.

HISTORY OF WORK

A) Updated Gantt Chart For Accomplish Milestones



Mario Run Project Schedule

GanttChartTemplate © 2006-2018 by Vertex42.com.

I0nics

Project Start Date 10/17/2019 (Thursday) Display Week 2

Project Lead

WBS	TASK	LEAD	START	END	DAYS	% DONE
1	**Phase 2**			-		
1.1	System Design Report	I0nics	Thu 10/17/19	Sun 10/20/19	4	100%
1.2	Physics Research	i0nics	Sun 10/20/19	Sun 10/20/19	1	100%
1.3	Screen Movement Research	I0nics	Sun 10/20/19	Mon 10/21/19	2	100%
1.4	Character Movement	Bikram	Sun 10/20/19	Fri 10/25/19	6	100%
1.4.1	Running	Bikram	Sun 10/20/19	Fri 10/25/19	2	100%
1.4.2	Jumping	Bikram	Sun 10/20/19	Fri 10/25/19	3	100%
1.5	Starting Screen	Bikram	Sun 10/20/19	Mon 10/21/19	1	100%
1.6	Main Menu	Bikram	Mon 10/21/19	Thu 10/24/19	4	100%
1.7	playerProgress	Bikram	Mon 10/21/19	Mon 10/21/19	1	100%
1.8	PausePane	Hung	Mon 10/21/19	Fri 10/25/19	4	100%
2	**Phase 2 Shop**		Tue 10/22/19	Sat 10/26/19	5	100%
2.1	Shop Main Menu	Hung	Tue 10/22/19	Tue 10/22/19	1	100%
2.2	Power Up Shop	Hung	Wed 10/23/19	Thu 10/24/19	2	100%
2.3	Character Shop	Hung	Fri 10/25/19	Sat 10/26/19	2	100%
3	**Phase 3 Levels**			-		
3.1	Background movement	Bikram	Sat 10/26/19	Mon 10/28/19	3	100%
3.1.1	Platforms	Hung	Sat 10/26/19	Sun 10/27/19	2	100%
3.1.2	Insert Character	Bikram	Mon 10/28/19	Wed 10/30/19	3	100%
3.1.3	Instructions	Bikram	Wed 10/30/19	Thu 10/31/19	2	100%
3.1.4	Power Ups Usage	Hung and B	Wed 10/30/19	Fri 11/01/19	3	100%
3.1.5	Background	Hung	Wed 10/30/19	Fri 11/01/19	3	100%
3.1.6	DevMode	Bikram	Wed 10/30/19	Fri 11/01/19	3	100%
3.2	Level 2		Sat 11/02/19	Fri 11/08/19	7	100%
3.2.1	Insert Enemy	Dallas	Sat 11/02/19	Sun 11/03/19	2	100%
3.2.2	Platforms	Hung	Mon 11/04/19	Tue 11/05/19	2	100%
3.2.3	Background	Hung	Wed 11/06/19	Thu 11/07/19	2	100%
3.3	Level 3		Sat 11/09/19	Thu 11/14/19	6	100%
3.3.1	Insert enemy	Dallas	Mon 11/11/19	Mon 11/11/19	1	100%
3.3.2	Platforms	Jiaxu	Tue 11/12/19	Tue 11/12/19	1	100%
3.3.3	Background	Jiaxu	Wed 11/13/19	Thu 11/14/19	2	100%
3.4	Level	bikram	Sat 11/16/19	Thu 11/21/19	6	100%
3.5	ScanLevel	Bikram	Fri 11/15/19	Wed 11/20/19	6	100%
4	**Phase 4 Finishing Touches**			-		
4.1	Collect Coins and Goomba movement	Dallas	Wed 11/20/19	Mon 11/25/19	6	100%
4.2	EndPane	Hung	Wed 11/20/19	Mon 11/25/19	6	100%
4.3	Character Collision	Bikram	Wed 11/20/19	Mon 11/25/19	6	100%
4.4	High Score	Hung	Fri 11/22/19	Sat 11/23/19	2	100%
4.5	Code Freeze GitHub	i0nics	Sat 11/23/19	Mon 11/25/19	3	100%
4.6	Final Presentation	i0nics	Sun 12/01/19	Sun 12/01/19	1	100%
4.7	Final Project Report	i0nics	Mon 12/02/19	Wed 12/04/19	3	100%

B) Comparison With Previous Milestones

Original (Report 1 & Report 2)	Updated
Phase 4 Shop(11/16/19)	Phase 2 Shop(10/22/19)
Additions:	ScanLevel(11/15/19)
LevelPane(10/22/19)	LevelPane(10/27/19)
Additions:	Level(11/16/19)
Additions:	endPane(11/20/19)
Instructions(11/20/19)	Instructions(10/30/19)
Additons:	CollectCoins and Goomba(11/20/19)
Inventory(11/19/19)	Deleted
Additions:	Collision(11/20/19)
Addition:	Collect Coins(11/20/19)
Additions:	Goomba Movement(11/20/19)
Additions:	High Score (11/22/19)
Additions:	playerProgress(10/21/19)
Addtions:	DevMode(11/30/19)

C) Key Accomplishments

- ScanLevel: The ScanLevel class helped us design multiple levels with ease, without having to hard code every image as a new GImage object. We achieved this by using a complex HashMap which consists of a string (level object name) as the key and an ArrayList of Pairs of two integers representing the x and y coordinates of the level objects. This class reads the file line by line and keeps adding object coordinates into the HashMap.

- Collision: Implementing collision was another challenging task that we were able to accomplish. By completing collisions, we were able to prevent the characters from running through the environment objects and allow the detection of coins and fluidly differentiate between the when the enemy's body collides with the character (kills character) versus when the enemy's head collides with the character (defeats enemy). In Super Mario Run, character and enemy collisions is checked for every 30 milliseconds using a timer. Furthermore the character consists of three hit-boxes for detecting different collisions. Head collisions help prevent the character from jumping up through a floating platform, body collisions help prevent the character from go through objects while running and finally feet collision to detect if there's ground under Mario. Feet collision occurs both during jumping and running so that the character automatically keeps falling down until an environment object is detected. It nothing is detected, it implies that there is no ground beneath and hence will make the character fall through and kill him.

- LevelPane: Finding a way to incorporate every class we used and put it all together into one level was a challenging task we had to take on. Our levelPane class was able to accomplish this by taking movements from character and goomba, while moving the environment as the user progresses through the level.

- Shop: When we first wanted to have a shop, we had to find a way to connect everything while saving the progress that the user made. PlayerProgress made it much easier to track how much progress the user has made in the game. By creating setters and getters for power-ups and characters, we were able to implement the characters and power ups into the level while keeping track of the number of coins the user has.

CONCLUSION & FUTURE WORK

BIKRAM CHATTERJEE

Real-world applications such as Super Mario Run are large, convoluted, and intricate that require a certain level of patience and skill to develop. Unlike my experience in my past computer science courses during which I was easily able to wrap up projects in no time by diving right in, this project gave me tremendous insight into developing real-world software which required me to put some actual sophisticated thought, planning, and structure before starting to write any code.

One of the biggest technical challenges that I encountered was forming the core structure of the program itself that involved solving the enigma of connecting related classes in a manner such that they seamlessly work together to perform a required function, and at the same time, preserve code maintainability. Designing the UI and UX was another extensive challenge as most of the graphics (images/gifs) of the original game were not readily available online. We wanted to steer away from the retro Mario graphics and move on to something that looked more modern. As a result, we had to adopt a painstakingly laborious process that involved taking screenshots and recordings of items directly from the original iOS game. Furthermore, we had to individually split them into 20-25 frames, crop out the unessential background from each frame, and finally place them in the game by calculating the required coordinates.

Some other challenges that I encountered involved figuring out the level's environment movement for a side-scroller, character jumping physics, collision detection, file reading to incorporate object location data into a complex HashMap data structure, creating a developer mode to make it easier for my team to place objects on to the level. The software engineering techniques that I learned, particularly in the interactivity, timers, and HashMap labs enabled me to address these challenges effectively. For instance, the timers lab which covered object collision helped me realize that creating multiple collision points to detect objects would result in higher preciseness. Similarly, the HashMap lab provided me the means to store an ArrayList of level object coordinates and conveniently access it by using the name of the object as a key.

An important feature that I was not able to finish implementing in the game is a level maker that allows both the user and developer to create custom levels without requiring them to manually type in the coordinates of level objects into a text file. I believe that learning how to work with JSON files would have provided me with the skill required to finish implementing this. I plan to continue working on this project in the future by possibly including levels that consist of moving platforms, question mark blocks, and different kinds of enemies such as Koopa Troopa, Piranha Plant, Boos, and Bowser. To conclude, I would like to advise future COMP 55 students to refrain from diving right into the project before proper planning, avoid programming your GIFs to resize as it is a performance-costly process that makes your program lag and start refactoring code at least two days before the due date.

HUNG LE

Throughout the project, I ran into many challenges along the way. Some were solved within hours while others took days to resolve. At the beginning of the project, I did not know how we were going to connect all the classes we had in order to make one level. As the project progressed, I started to have a better understanding of inheritance and how classes interacted with each other in order to connect different classes. I started to use methods from classes like playerProgress in order to make and save transactions within the shop. Debugging was also a big problem for me. Little things like coins being counted and added incorrectly took longer than I expected to spend on such a simple task. I had to set breakpoints in certain parts of my code in order to see where the error was coming from. Past projects like traffic jam gave me a strong starting point in order get myself ready for bigger project like Mario. For future plans, I will help develop more levels as well as adding more enemies to each level. Increasing the length of each level while adding more platforms, coins, question mark blocks are all things I can work on in the future. A piece of advice that I would give to future comp 55 students is to focus on the backend of the game rather than putting a lot of effort into the graphics. Getting the basic functionality of the game is much more important than how the game looks.

JIAXU TIAN

When I was working on the program, one of the most serious technical challenges I encountered is debugging. One of the most important jobs for me in the last week is to test the whole program. When I find a bug in the program, how to do the debug is becoming a challenge to me. However, I remembered the software engineering techniques that I learned in debugging lab. Setting a breakpoint is really helpful to me when I use the debug mode, and ask other teammates is really helpful. There are also some other software engineering techniques I learned in class are helpful for this project. I think most of the lab I did was very helpful to me, and we use the board and space from the vehicle lab. For me, the future work of this project is to fix the naming part of this project. I did not realize this problem until professor point that out, and it is a simple problem which we can totally avoid. My advice to future students is about time management. We had a bigger plan for our program in the beginning, but our product does not have all the functions from the plan. I think if we did a better job at time management, we could build a product exactly the same as the plan.

Dallas Reynolds

During this project I faced a lot of challenges. Most of these challenges came from my lack of skill in programming and trying to figure out what to do on the project. The software engineering techniques I learned that helped me the most with this project were the debugging lab and the GitHub lab. These labs helped me with making sure the code that I did write was working and that I was able to send it to the rest of the group. After working on traffic jam I was more comfortable with writing code and working on something larger than 500 lines of code. I wish I had more knowledge about some of the code that my fellow group mates were writing. That would have helped me keep up better. I learned that it takes a lot of thought and time to complete a project of this size. This has definitely increased the amount of respect I have for people that work on large projects. For future work on this project I would learn more about different programming tools whether it be the code itself or knowledge on how eclipse works. If there would be one piece of advice that I would give to a future Comp 55 student it would be to start on planning the project as soon as you can and don't let yourself get behind or it will make it much harder to catch up and figure out what is going on. Also maintain contact with your group because this is not an easy task to complete if you aren't an experienced programmer.

REFERENCES

- https://www.mariowiki.com This is essentially the wiki guide for Mario. It provides instructions on how the game works and contains a huge database on marioverse.

- https://supermariorun.com/en/ This is the website of the original Super Mario Run game which is also the original inspiration for our game.

- https://www.nintendo.co.jp/clv/manuals/en/pdf/CLV-P-NAAAE.pdf This is a manual for one of the original Mario games which contains vital information about how the game works.

- “Super Mario Run Revenue and Usage Statistics.” Business of Apps, 27 Jan 2017, www.businessofapps.com/news/super-mario-run-revenue-and-usage-stats/. This is an image that shows the longevity of a games like Mario Run vs Pokemon Go.

- https://www.statista.com/chart/7344/nintendo-stock-price-super-mario-run/ Statisa is where we found the statistics on the money that Mario Run grossed and how it fared against other games.

- https://en.wikipedia.org/wiki/List_of_most- played_video_games_by_player _count This wiki contains a list of the most played video games by player count. This is evidence to the type of games people are interested in.

- https://www.thecrydsdaily.com/p/super-mario-run-requirements.html This is where we got the comparative hardware requirements for the existing Super Mario Run.

- https://www.macworld.co.uk/feature/mac/os-x-macos-versions-3662757/ Here is where all of the current functional Mac OS versions can be found.

- https://en.wikipedia.org/wiki/Windows_10_version_history Here is where to find all of the current and past Windows OS.

- https://stackoverflow.com/ This is a great website for programming help and examples of code that might help with a future project.

- https://www.youtube.com/watch?v=2dZiMBwX_5Q This is a YouTuber named Caleb Curry. If there is something that might need refreshing this is a good source for information on coding in Java.

- https://www.geeksforgeeks.org/parse-json-java/ To build a level maker, we should have incorporated JSON files into our project. This website can provide insight into JSON file implementation to programmers who want to build on our project in the future.

- https://cs.stanford.edu/people/eroberts/jtf/javadoc/student/acm/graphics/GRectangle.html This website helped us understand more about GRectangles in order to implement collision detection. We found out that using GRectangles is far more efficient than multiple plain collision points as it required less work and it was more effective at the same time.

- https://play.nintendo.com/activities/play/super-mario-maker-desktop-background-wallpaper/ This website provided us with amazing level backgrounds from the original Mario Bros U game.

- https://www.kapwing.com This website helped us develop the menuPane background which took almost 3 days to make! Kapwing helped us solve the challenge of taking 3-5 separate screen recordings from the original iPhone Super Mario Run game and then put them together side by side so that it creates the illusion of having one desktop application size video instead of 5 separate iPhone screen recordings.

- https://ezgif.com We spent a lot of time to crop out the background of almost 5-24 frames/pictures of each character running and jumping. After finishing this time consuming process, ezgif helped us convert these pictures into a gif in a matter of seconds. Unlike other gif converters, it also has a coalesce mode which takes out the bad frames which java is incapable of handling.