MAD ROCKET GAMES

Design Documentation

Ready, Steady, Roll!

Written by Karl Inglott

TABLE OF CONTENTS

README

This document should be read by all team members. It is important that anything that is developed for the game is developed under the guidance of this document. If there are aspects of this document that need further details or are not fully understood, ask your funky designer Mr Karl Inglott to confirm, elaborate or explain features as required.

This document should be used to better understand the functionality, scale and processes required for the game Ready, Steady, Roll! To use this document, look in the contents page for the required section and click the heading. Generally, each section is divided into subsections which further break down components. Sections will reference other sections which are directly relative to component functionality. These are all hyperlinked within the document and can be clicked to move to that section. It is advised that any section linked is read before work has started on that section.

This document is a constant work in progress. Whilst it will be as comprehensive as humanly possible, it is always possible that details or sections may be unintentionally overlooked. If there are any problems, as mentioned above, speak to the designer and author of this document, Karl Inglott.

As this document is a work in progress, details and sections may indeed change over the course of development. Any changes should be monitored so that team members are in tune with the project direction. Any changes will be listed in the Version History.

If any changes are made to the design document, it is required that whoever makes said changes, writes information of them in the version history. This includes the version number, the date, the description of changes and who made the changes.

Note: This document has been formatted with the MoSCoW prioritisation technique for all relevant aspects of the game. This is noticeable in the headings in the contents page, by which most headings have an ***M, S, C*** *or* ***W*** *suffix applied. As a brief description of MoSCoW, the acronym stands for:*

- *Must have (M)*
- *Should have (S)*
- *Could have (C)*
- *Won't have (but would like to) (W)*

Use this as a simple guide to where the importance lays in the project development.

VERSION HISTORY

The version history lists all changes that have been made to the design document.

Version	Date	Description	Changed by
v.0.1	20th May 2012	Design Document Inception: Partial Content Headers Included and document formatting completed.	Karl Inglott
v.0.2	30th May 2012	Sections fleshed out. Diagrams for Galaxy View and Game Flow	Karl Inglott
v.0.3	31st May 2012	Partial Naming Conventions detailed.	Karl Inglott
v.0.4	2nd June 2012	Mechanics detailed, networking detailed, game states detailed, audio briefly detailed, assets and naming conventions further detailed	Karl Inglott
v.0.5	4th June 2012	Audio fleshed out, AI detailed, Cameras detailed added overall further details. Added new images for reference for main game camera, in game GUI and front end. Added Achievements section. Added more information on World and Setting. Race tracks briefly detailed.	Karl Inglott
v.0.6	8th June 2012	MoSCoW applied to headings, layout of sections modified. Race Tracks expanded with track segments. General modifications of mechanics made. Achievements expanded with achievement notifications Added a readme section. Moved the world and setting section. Expanded the Balls section	Karl Inglott
v.0.7	9th June 2012	Added in Scenes section listing scene transitions and loading screens. Reworked the front end section for carousel idea. Fleshed out audio and character animations. Fleshed out audio and animations in the asset list. Added in-game debugging section	Karl Inglott

GAME SUMMARY

HIGH CONCEPT

Ready, Steady, Roll! The must have multiplayer racing game for Android tablets! Tumble your way downhill on a giant ball of cake, squashing into and collecting delicious ingredients to build up your size and speed! In a bid to find the answer to life, the universe and everything; help the makers and bakers build a galaxy out of cakes. Compete against your friends online for a scrumptious, mouth-watering experience which will leave you asking for more. The cake is not a lie.

GENRE

3rd person multiplayer racing.

PLATFORM

Primarily Android tablets, although it should be compatible with iOS.

UNIQUE SELLING POINTS

- Multiplayer Racing on Tablet devices
- Momentum based racing
- Food Themes

VISUAL STYLE

Friendly colours with an imaginative food themed world of ice cream trees and doughnut ring tunnels. Objects as small as a football and so large you cannot see the top.

AUDIAL STYLE

Funny and quirky. The sounds must complement the cartoony visuals and the food and give the player comedic and fun feedback to the gameplay. The music must be reflective of the setting and must keep up the pace with the fast gameplay.

TARGET MARKET

The focus in on ages 8 and upwards, though aspects appealing to an older age demographic. It has a geographically wide market due to the lack of in game text which would need localisation.

SCOPE

- 4 Characters
- 3 Tracks
- Lots of environment objects
- Single player and Multiplayer
- Interactive Leaderboards

GAMEPLAY

The gameplay is centred around racing and momentum. This section will detail the gameplay in full, explaining how the gameplay should function in the end product.

SUMMARY

The player is in control of a tumbling ball of goodness that travels down cylindrical tracks. The player is able to turn left and right and must collect objects down the track to increase momentum. The player can crash into objects and other players to lose momentum. The player can also use the terrain to his or her advantage by using ramps and taking different routes.

CONTROLS (M)

The controls are fairly simple, this section will cover the controls. The controls comprise of:

- Touching objects for navigation
- Tilting Device left/right for movement
- Swiping Motion left/right for rotation around a fixed point

MENUS AND FRONT END (M)

The front end and menus will require single button/object presses by touching the button/object on the screen. There will be two triggers:

- On down – Play a sound
- On release – Play a sound and perform button commands (move to different menu, play the game etc)

GALAXY VIEW (C)

Galaxy View, similarly will require touch gestures to interact. This will be in the form of objects that can be touched to zoom in and out from them. The view can also be rotated by swiping left and right along the screen.

- Planets are selected on release of touch gesture
- Rotating should prompt a sound
- Selecting a planet should prompt a sound

IN GAME (M)

The interactions in game comprise of only the tilting action to move the player left and right. The tilt should work as detailed in the below diagram.

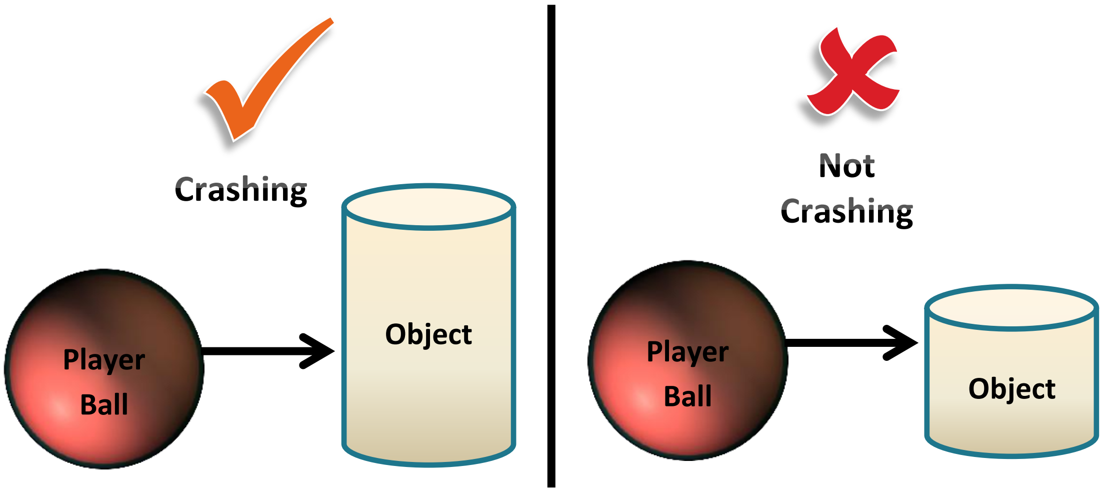
- It should check the angle of the device before racing begins, and set that tilted angle as 0.
- If device is tilted for longer than 4 seconds in same direction, it should reset the tilting value to 0 at that tilted angle. (As a way of allowing players to change position whilst playing)

CRASHING (M)

Crashing causes several events to occur. Crashing occurs when:

- Player hits object taller than the ball
- Player hits other player

The size is in reference to the height, as can be seen in the below diagram.



Players who crash should drop parts of their mass on the track that other players can collect.

OBJECT (M)

If the player crashes into an object:

- The ball will lose 5% momentum
- The ball will lose an equal amount of mass
- This percentage should increase depending on the size (the larger, the more unstable the cake ball)
- The ball will ricochet from the object
- It will trigger a crashing particle effect, colour and sprite dependant on the collided object
- It will trigger a sound reflective of the crash
- It will trigger a sound from the character
- It will trigger a crash animation from the character

PLAYER (M)

Crashing into other players will yield the same results although there are other factors to consider.

- Smaller players can crash into larger players without the momentum loss
- If two or more players collide, it should consider which players swerved into who (S)
- A player swerving into another should cause the non-swerving player to lose momentum
- If both players swerve it should cause both to lose momentum
- If two players swerve into another player, too bad for the non-swerving player
- Swerving should be checked based on sudden jumps in the x axis of the tablet (S)
- Character sounds and animations will differ to crashing into an object

MASS AND MOMENTUM (M)

Momentum will be dependent upon the Mass. The larger the cake ball, the more momentum it should be able to garner.

- Crashing into smaller objects will increase Mass gradually for a set period
- Mass will increase constantly as it collides with the track
- After crashing into a smaller object, the object should scale on its local y axis so that it looks squashed (S)
- Object should be visibly attached to the ball before gradually disappearing (S)
- After object has been absorbed, it should leave a texture in the place of collision. (S)
- Particle effect should trigger from object crash reflecting the object colour and type
- Sound should trigger reflecting the type of object absorbed
-

JUMPING AND MOVEMENT (M)

Jumping occurs when the player moves off either a ramp or hill.

- Distance is dependent on mass
- Rolling sound is discontinued for a whooshing sound
- Player animation should change to a flying through air animation
- On impact with the ground, a blast radius three times the radius of the ball will occur around the player (C)
- If any players are within the players radius Blast radius it will have the same effect as player collision (C)
- On impact with the ground it should trigger a crashing sound

TUNNELS (S)

Tunnels should allow the player to tilt and roll around on the tunnel walls.

- If player stops tilting whist on the walls, the ball should roll back down to the ground level
- If the player exits the tunnel via the roof, the player should slowly turn to right position before landing

SLIPSTREAM (C)

Slipstream works as with most racing games.

- Slipstream area is identified by bald trail on the track that is left behind by the player(s) ahead
- A particle effect should float above the slipstream area, like a cookie trail
- A player entering the slipstream area should have speed increased by a set amount
- If a player leaves the slipstream, speed will be returned
- Entering a slipstream should trigger a character sound “wooo”

"GRAVITY" & MOTION (M)

- Players should be in constant motion down the track
- Players should be able to move around the surface of the track
- Moving off a sudden hill or ramp should cause player to "jump" (see Jumping And Movement)
- Players can not fall off the track until the end
- Segments will vary in radius
- Segments will have objects on the surface that can be rolled on top of

IN-GAME STATES (M)

There are 6 possible states whilst in a game. Three of these states occur within the race state itself. These states have specific requirements which are listed in each of the headed sections below. The states are:

- Pre-Race State
- Race State
    - Countdown State
    - Menu State
    - Follow Player State
- Post-Race State

PRE-RACE STATE (M)

The pre-race state occurs once the player(s) have selected the track and characters and the assets have loaded. This state will show a skip-able animation sequence of the characters of them preparing themselves for the race.

- State is skip-able if any players touch the skip button located on the screen.
- The sequence should be short
- It should feature all of the players competing in the race

RACE STATE (M)

The race state is the racing aspect of the game. The players will be able to:

- Turn left and right
- Collide into each other and the environment
- Collect objects to build mass
- Open the menu
- When each player reaches the finish line it should trigger a short, fanfare sound
- When each player reaches the finish line it should also trigger a happy creature sound

The state also has three states contained within it.

COUNTDOWN STATE (M)

The countdown state is just before racing begins. It is for the player to ready themselves for the race.

- After 1 second of the state fully loading, a 3 second timer will countdown to 1
- The timer will be visible on the screen
- A sound will trigger at each second
- At 0 seconds, a sound will trigger signalling the players to go
- Go! Will also be displayed on the screen at this point
- The camera will be the same camera used in the race state
- The player can interact with the camera as they will in the race state
- The player can open the menu

MENU STATE (M)

The menu state will be triggered if the menu button is pressed in the race state.

- The menu state will pause the game in a single player race
- It will not pause the game in a multiplayer race, and the player pressing it will continue moving forward
- The menu state will show the in-game menu
- The player can still control the movement of the players character by tilting
- The player cannot interact with the camera movement
- The player can quit the game
- The state will end if the player quits the game or if the player presses the resume button

FOLLOW PLAYER STATE (M)

The follow player state starts when a player has finished the race but there are still other players racing.

- The player begins spectating the player in last place
- The player has control over his/her “spectating” camera
- The player has no control over the spectated players camera or movement
- The player can open the menu
- The state will end if all players have finished

POST-RACE STATE (M)

The post-race state starts once all players have finished the race. It will be a skip-able animated sequence.

- State is skip-able if any players touch the skip button located on the screen.
- It should feature all of the players competing in the race
- It should visibly show what place each player came

Post-Race Stats

The post-race stats screen will show how each player performed. It will display:

- What position each player finished in.
- The times of each player
- Who collected the most objects
- Who crashed the most
- Who was the biggest
- Who made the most jumps
- Touching the screen will send the player back to the lobby

SCENES

Scenes are what Unity uses to differentiate between different parts of the game. We will require 6 scenes:

- Front End
- Pre-Race
- Post-Race
- Level 1
- Level 2
- Level 3

SCENE TRANSITIONS

Scene transitions will occur when the player moves through different scenes.

- Screen transitions out of a scene should be a fade into black
- Screen transitions into a scene should be a fade from black
- Each way should last no longer than 1 second

LOADING SCREENS

Where loading is required between scenes, it should display a loading screen for the player to watch in awe and amazement.

The loading screen should display help screens on how to play

- One screen should show how the player should collect smaller objects and avoid larger ones
- One should show that collecting objects increases mass and therefore momentum
- One should show that the player must tilt the device to turn
- It should have text stating that it is a loading screen

If possible, each of these screens should show in one load. If it’s found that they do not display for long enough, it should randomly select one to show at each load.

WORLD AND SETTING

CHARACTERS (M)

There are 4 characters to be made for the game. Each character should be roughly the same dimensions for ease of rigging and animating.

CHARACTER ANIMATIONS (M)

Each character will have a set of animations to be used in game. These animations are divided into gameplay animations and scene animations.

GAMEPLAY ANIMATIONS (M)

There are 18 gameplay animations for each character. Whilst these animations can differ for each character, this is not paramount. Each character has three sets of animations for three levels of speed. Each of these animations should be a seamless loop for use in the game. The types of animations needed are:

- Running
- Gestures Whilst Running
- Colliding
- Gliding through the air
- Idle

SCENE ANIMATIONS (M)

The scene animations refer to animations that play as part of a non-interactive sequence or cutscene. These include the pre and post-race sequences. These scenes must be in game with the animations made in Maya.

PRE-RACE SEQUENCE

The pre-race sequence should follow the cliché of some racing tv show/games where it has close ups of the character putting on racing gear. This instead of a helmet, gloves and boots should be an apron, a bakers hat etc. All of the animations should be made in Maya and then the camera work done in Unity.

POST-RACE SEQUENCE

The post-race sequence should show the balls racings towards the star at a distance and finally coming to a stop. It should show the winner ball stopping first, with the other balls appearing in order of where they finished the race, and each getting caught in the gravity of the winner. It should show them all jumping up and down on the planets/moons they created.

BALLS (M)

Each player controls a ball to navigate the levels. Characters stand on top of their balls and run to mimic the control and movement of the ball.

- Balls roll in reaction to the momentum of movement and the turning
- Colour of the ball starts at a default colour
- The colour changes as the player progresses through the track
- Balls should have a texture applied depending on objects collected (C)
- Balls increase in mass as they collide with the track and with merge-able objects
- Balls can collide into objects and players (see Collisions)

ENVIRONMENT (M)

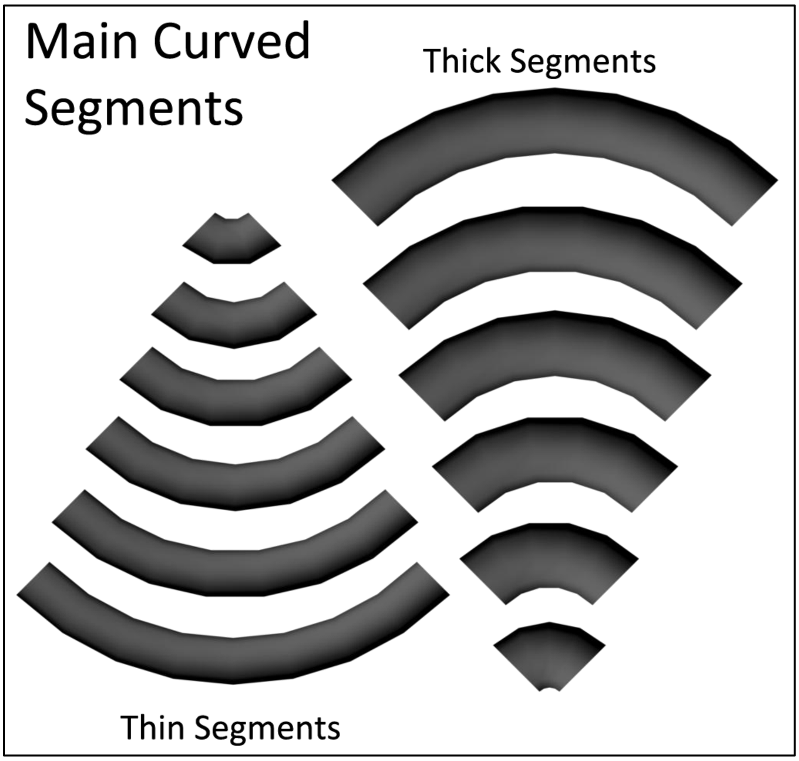
The environment is made up of several elements.

- Track Segments
- Terrain Objects
- Absorption Objects
- Players (see Characters)
- Background
- Lights
- Particle Effects

TRACK SEGMENTS

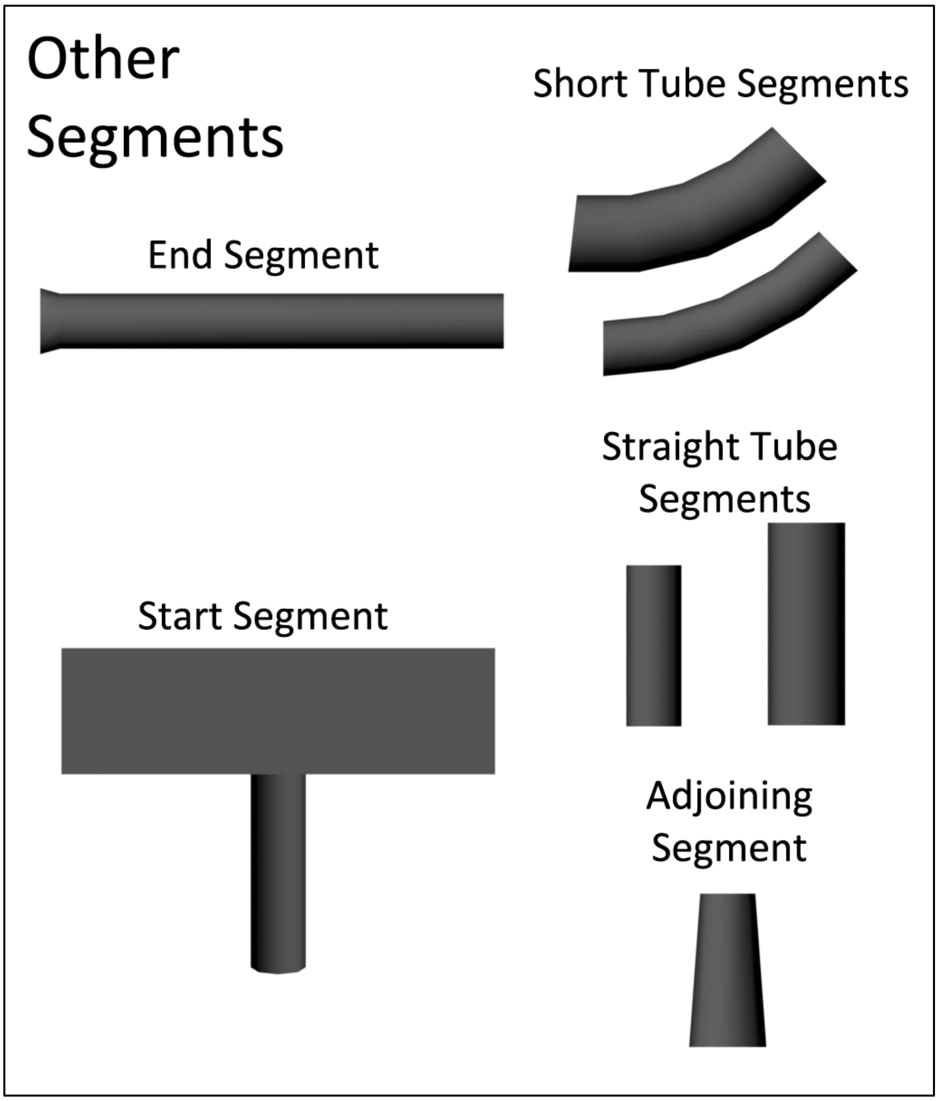
The race track segments should be made up of 17 objects. Each of these objects should be attachable so that tracks can be streamlined and modular. The main bulk of the objects comprise of the main tube curves

The main segments are ¼ of a fully circular tube and increase in scale by 1. They are also divided into thin and thick segments which can be joined up with an adjoining segment.



TRACK SEGMENTS CONT…

The other segments are made up of two shorter curves which will allow for a less rigid style of track. It also includes the start and end segments, straight segments for both thicknesses and the adjoining segment allowing for the transition between the thick and thin segments.



TERRAIN OBJECTS

Terrain objects are aspects of the tracks that the player can roll into, or over. These segments are applied through unity onto the track segments. The types of objects required are:

- Hills
- Mountains
- Milkshake water
- Tunnels

ABSORPTION OBJECTS (M)

Objects in the game can be absorbed into by the player. These include several different types of object.

- Ice Cream Trees
- Candy Floss Trees
- Strawberries
- Cookie Rocks
- Doughnuts
- Ice Cream Rocks
- Coconut Mushrooms
- Muffin bushes
- Lollypops

BACKGROUND

The background is the background of the game world. This includes:

- Skybox
- Planets
- The Star
- Floating objects

Background objects are not able to collide with the player and should be at a safe distance away from the potential player track.

LIGHTS

Lights will be used within the environment. Lights will be used in the pre and post-race sequences for emphasis and will be used to illuminate the game world.

- The nearby star will need to have a light source
- The star will have a lens flare effect
- There will be an ambient light illuminating the game world

PARTICLE EFFECTS (S)

Particle effects will be used in many instances and are listed throughout this document. The instances where particle effects will be required will be:

- Moving over terrain
- Collecting objects
- Flying through the air
- Hitting the ground
- Player Collisions
- Terrain Object Collisions
- Achievement Notifications
- Moving through water
- Post-Race Sequences

Each of these will have different properties and various textures.

AI (C)

AI will be required if less than four human players are racing. AI should follow splines or paths which can be defined in Unity by placing markers.

The AI will be affected just as human players are by gravity, collisions, jumping and slipstreams.

SPLINES OR PATHS

- Each AI has a separate path
- The splines or paths should define the route for the AI
- AI should move to the next marker in their path
- Markers are not physically visible in the race and only visible in Unity
- AI should smooth between markers and not just instantly flip into the correct position for the next marker
- If the AI is behind for more than 6 seconds, the AI should look for the closest slipstream and use it as the path until the AI gets within a close distance to the player in front

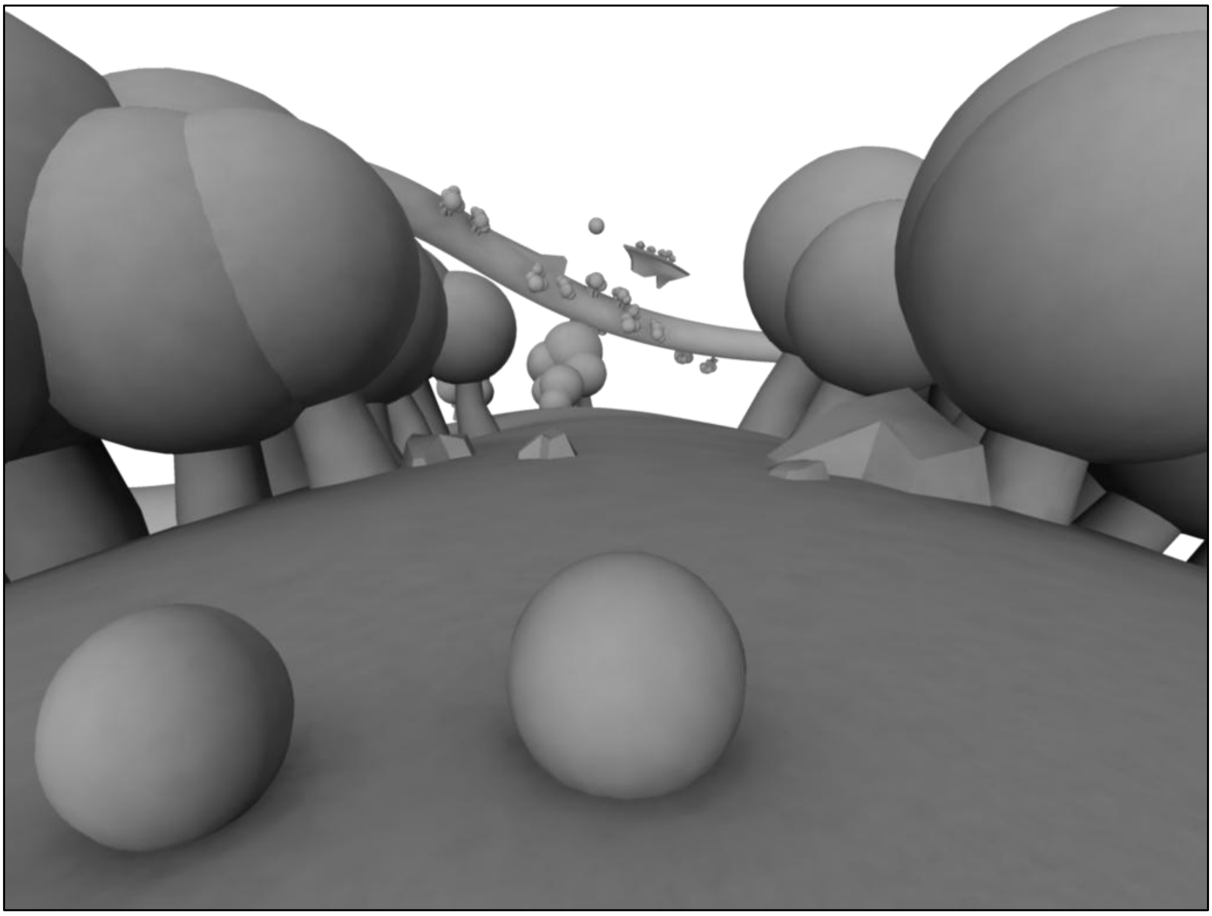
CAMERAS (M)

Cameras at different points in the game will behave differently. This section will detail how cameras should function in each differing instance.

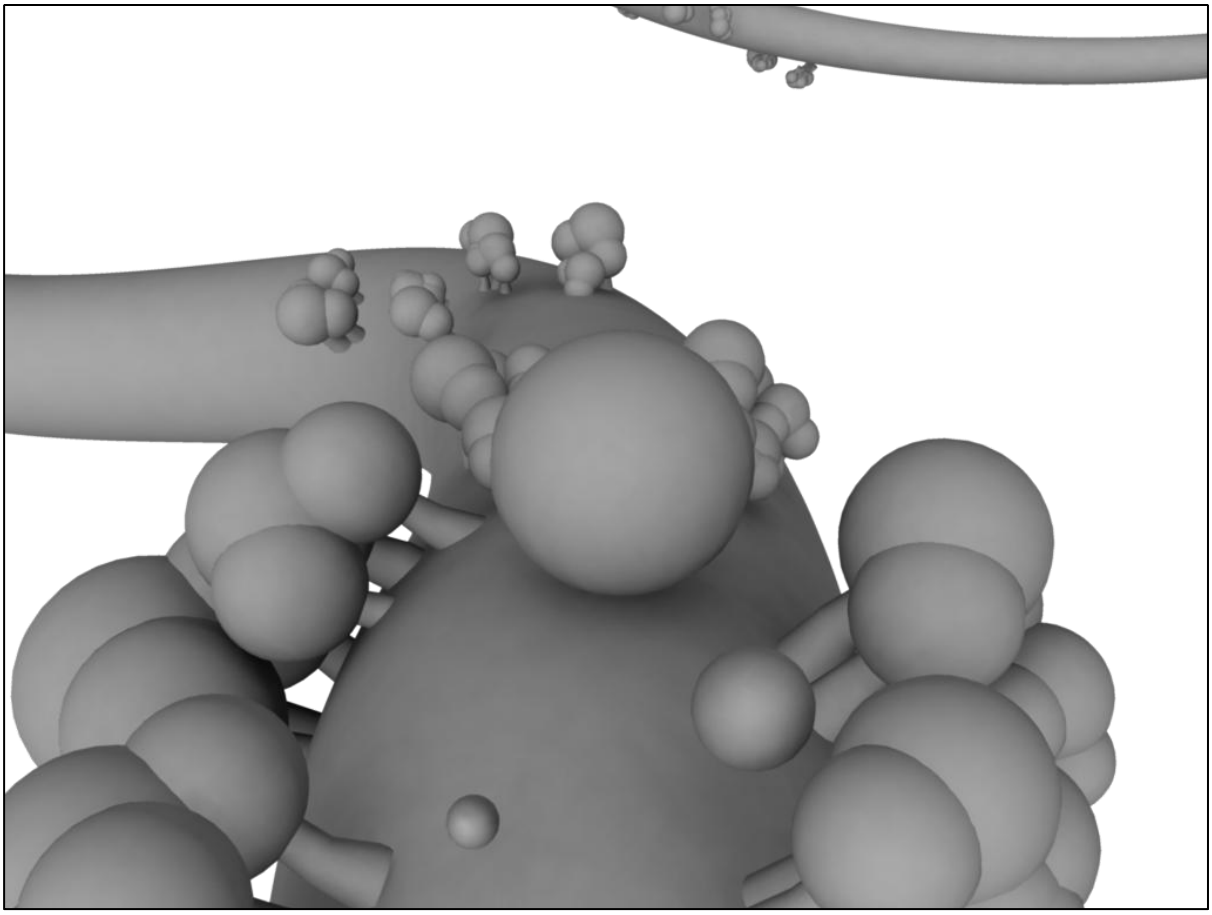
MAIN GAME CAMERA (M)

The main game camera is the camera used whilst racing. This camera will follow the player from an angle above the player.

- The camera always follows the player
- The camera will have smoothing applied for when the player turns
- It should increase the field of view as the player builds up momentum
- It should apply a motion blur effect which intensifies as momentum builds up (C)
- This camera should be in use just before the initial race start timer and just after each player reaches the finish line
- If another player appears behind the player, the camera should smoothly zoom out to give a better view (S)
- Camera should zoom back in when the player behind has moved away (S)



MAIN GAME CAMERA CONT…



DRAW DISTANCE

The draw distance should allow the player to see all future segments of the tracks but does not need to see all of the objects. The player should be able to see all of the skybox contents.

GALAXY VIEW CAMERA (C)

The galaxy view camera will change depending on interaction.

- It will start fixed to the star
- It can be rotated around the star's origin
- It will switch to objects (planets & tracks) selected on the screen for a side on close up view of the planet
- It will smoothly build up speed as it approaches the planet/star/track and smoothly slow down before it reaches it
- The player will not be able to rotate it if it is fixed on a planet
- When moving focus from and to the star, it should change the field of view dependant on the movement speed of the camera

MAIN MENU CAMERA (M)

The main menu camera will show the main menu and title screen.

- It is set at a fixed point in the scene
- It can zoom in and out of parts of the screen on button selection, depending on the button selected
- The zoom must be smooth

PRE-RACE & POST RACE CAMERAS (M)

Both the pre-race cameras and pre-race cameras will move in a scripted way.

- There are several cameras
- Cameras are non-interactive
- Cameras can switch between each other
- They will move according to a script

AUDIO (M)

Audio in the game is used in all areas and is an important aspect of the game. This section will cover all required audio for the project. Some audio will be 3d, meaning it will have a radius and only the players within that radius can hear it. This will include collision sounds or character sounds.

- All audio should have no preceding or succeeding silence to enable instant audio playback and looping.
- Audio should be stereo
- All audio should maintain a consistent gain threshold across all files
- All audio files should follow the set naming conventions
- All final audio should be in the .AAC file format

Sound assets lists and folder directories are located here: Assets

MENU SOUNDS (M)

Menu sounds are made up of the button feedback sounds for interacting with the menu, character sounds for selecting a character and animation audio for the background animation. They should be fun and in line with what's going on. They should also be relatively snappy.

BUTTON/UI FEEDBACK SOUNDS (M)

The button feedback audio is used on different buttons in the menu system.

- Positive On Down (sfx_fe_posdown)
- Positive Release (sfx_fe_posrelease)
- Negative On Down (sfx_fe_negdown)
- Negative Release (sfx_fe_negrelease)
- Change Selection (sfx_fe_changesel)
- Completed (sfx_fe_complete)
- Exit (sfx_fe_exit)

Major buttons are buttons that change between menu screens. Minor buttons are buttons that change options in menu screens. On down refers to when the finger is pressed down, release refers to when the finger is released.

CHARACTER AUDIO (M)

The menu sounds will also make use of character audio. Each character will have a sound clip which will play when the character is being viewed. This will include:

- Character 1 Sound (sfx_fe_char1)
- Character 2 Sound (sfx_fe_char2)
- Character 3 Sound (sfx_fe_char3)
- Character 4 Sound (sfx_fe_char4)

BACKGROUND AUDIO (M)

Some of the background audio will be tailored for the animations in the front end. This audio will be made up of mechanical sounds. Audio that does not need to be tailored is:

- Rolling pin conveyor belt movement sound (see Front End Visuals & Functionality)

IN-GAME SOUNDS (M)

In game sounds refers to all sounds that occur when playing in a race. These sounds include:

- Collision sounds
- Feedback sounds
- Environment sounds
- Character sounds

COLLISION SOUNDS (M)

Collision sounds occur when the player collides with other players or with objects in the environment.

There will be several collision sounds required which will be linked to their respective object.

- Ice Cream Tree (sfx_env_icecreamtree)
- Candy Floss Tree (sfx_env_candyflosstree)
- Player (sfx_env_player)
- Cookie Rock (sfx_env_cookierock)
- Strawberry (sfx_env_strawberry)
- Coconut Mushroom (sfx_env_coconutmushroom)
- Doughnut (sfx_env_doughnut)
- Milkshake (sfx_env_milkshake)
- Lollypop (sfx_env_lollypop)
- Ice Cream Rock (sfx_env_icecreamrock)
- Apple (sfx_env_apple)

RACE FEEDBACK SOUNDS (M)

Race feedback sounds occur when something happens in the game that requires some audible feedback.

- Countdown (sfx_fbk_countdown)
- Achievement Notification (sfx_fbk_achievement)
- Race Started (sfx_fbk_racestart)
- Race finished (sfx_fbk_racefinished)
- Player left the race (sfx_fbk_playerleft)

MOVEMENT EFFECT SOUNDS

These are linked to character movement

- Rolling loop (sfx_eff_rolling)
- Fly through the air (whoosh sound) (sfx_eff_flying)

CHARACTER SOUNDS (M)

Character sounds are the "voices" of the player characters. Characters should have an illegible language that reflects each character's visuals. Each clip should be fairly short, no longer than 5 seconds. These will trigger at specific events. Each character will have a total of 10 sounds. Per character, these sounds will be:

- 3 positive sounds
- 3 negative sounds
- Race start sound
- Race finish sound
- Flying through the air sound
- Front end sound

Information on the above sounds:

- Positive sounds will trigger when a player hits an absorbable object or gets an achievement.
- When absorbing an object, positive and negative sounds should only be trigger-able after 5 seconds of not hitting any objects and should randomly choose one of the three sounds available
- Negative sounds will trigger when the player collides into another player or an object.
- Flying through the air will trigger when the player starts travelling through the air after jumping from a ramp.
- The Race start sounds will trigger when players start the race
- The Race finish sound will trigger when the players finish the race
- Front end sounds will be tailored for an animation in the front end

MUSIC (M)

The game will require 5 full audio tracks. Each of these should be unique, but maintain an amount of consistency with the themes in the game. The music can be broken down into front-end music and in-game music.

FRONT END (M)

The front end will require two music tracks.

- Menu Screen Music (m_menu)
- Galaxy View Music (C) (m_galview)

MENU SCREEN (M)

The menu screen music must be upbeat and quite varied. It should keep the player interested and get the player in the mood to play the game.

- The menu music will play as soon as the title menu loads up.
- It should be between 1-3 minutes in length.
- It does not have to be loop-able.

GALAXY VIEW (C)

The galaxy view music should be atmospheric and ambient whilst still maintaining the consistency with the other tracks. It shouldn't have any abruptness to it.

- The galaxy view music will begin playing as soon as the galaxy view loads up.
- It should be between 1-3 minutes in length
- It does not have to be loop-able.

IN-GAME (M)

For the in-game music, 3 music tracks will be required. One for each of the race tracks. Each of these tracks should build up, following the themes of building momentum. They should either be fast paced from the offset or gradually pick up pace.

Each song must have a loop-able segment at the end of the track. This loop-able segment should be 30seconds approx. This means that the pace of the music will continue even if the players are sluggish getting to the finish line.

- There should be a track for race track 1 (m_track1)
- There should be a track for race track 2 (m_track2)
- There should be a track for race track 3 (m_track3)
- The in-game music will play as soon as the players begin moving (after the race countdown and pre-race intro)
- It should be about 3 minutes in length
- The final loop-able segment should be about 30 seconds in length and should be a separate music file.

FANFARE (M)

There should be 4 short fanfare songs. Each of these will reflect 1st through to 4th place finishes. The 1st place fanfare should be celebratory the 4th place fanfare should be doom and gloom. The fanfare will play directly after each player finishes the race.

POST-RACE MUSIC

There should be one track for the post-race. This should be a 20 second loop. It will play as soon as all players finish the race and after the fanfare.

RACE TRACKS (M)

The game should feature 3 tracks. Each track should be varied in shape, difficulty and objects contained within. Tracks should be made up of a factory located at the top of the track, this is the starting position, the main racing track and the finishing area. The tracks will be slalom style courses, but tubular in shape, with the player racing around the outer surface.

- Tracks are made up of tube segments that must be attached to each other in unity.
- Each track piece should have a symmetrical polygonal shape around the circumference along both the x and the y axis
- All environment terrain (hills, mountains) should be separate objects applied to the track mesh in unity.
- There should be a total of 19 track segments which can be attached in various ways
- All tracks should be around 3-4 minutes in length
- Tracks should be littered with objects for the player to collect or collide into
- Tracks should have a main route that sometimes branches off into multiple alternate paths
- Tracks require an amount of turning to avoid obstacles, especially towards the end

TRACK 1 (M)

TRACK 2 (S)

TRACK 3 (C)

NETWORKING (M)

As a multiplayer game, networking has a profound importance in the game.

SERVER (S)

Data will need to be stored and retrieved from a main server. This server will handle all of the data for setting up races and for leaderboards. The data for handling servers will be:

- Latency (latency should be checked on all players to find the most suitable host)
- All player IP's (sent by host when setting up a game, retrieved by potential clients trying to connect)
- Each Client Game Slots Remaining (Data sent when client connects/disconnects, retrieved after finding host IP)

LEADERBOARDS (C)

The data for leaderboards will be:

- Player Name
- Unique Player ID (Assigned per Facebook account or by Player Name)
- Planet Name
- Ball Size
- Race Time
- Race Track

It should be synced to the server at the end of the race.

CLIENT/HOST (S)

The client/host networking will be used to handle the in game races.

FINDING THE HOST (W)

The game should find the host by checking the players in the lobby

- It should wait until at least two players are in the lobby.
- It will test all player latencies and make sure hosting is possible with one or more devices.
- The selected host will send his/her ip to the server
- The clients will then retrieve the hosts ip for connecting
- If more players join once a host has been selected, it will continue with that host

HOST MIGRATION (W)

The host should migrate if:

- The host quits or disconnects from the game
- Latency with current host is too high

If this occurs it should follow the same procedures as when finding the host initially:

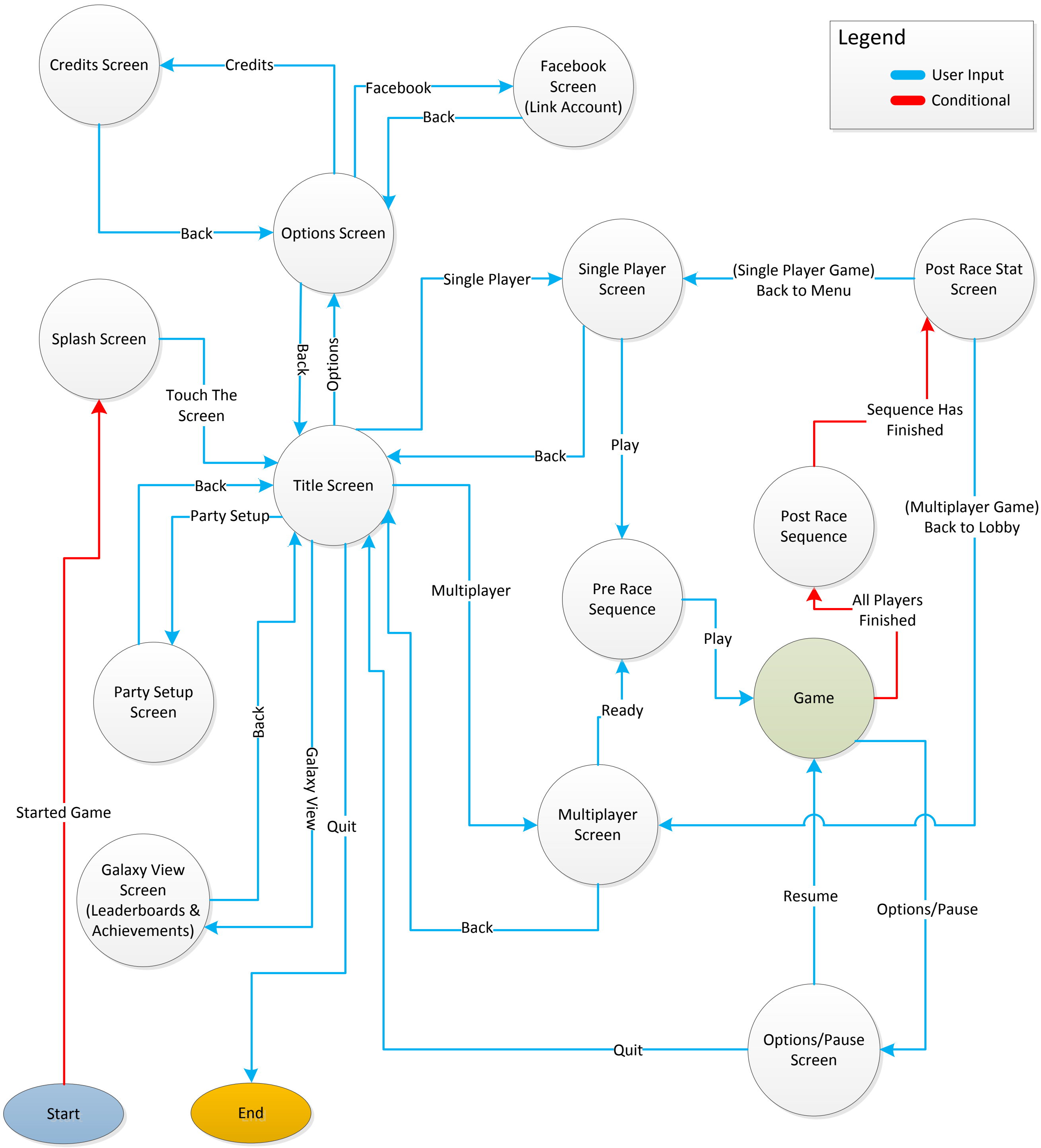
- It will test all player latencies and make sure hosting is possible with one or more devices.
- The selected host will send his/her ip to the server
- The clients will then retrieve the hosts ip for connecting
- Whilst it is looking for a new host, the players should still be in control of their characters.
- Other players should be seen to continue in the direction they we're moving before the host left
- Once a host has taken over, players should update their positions from their local device to the other players

If no suitable new host is found, the game should quit displaying an error message stating that the Racing host has left the game. Requiring each player to press okay, it will then return the remaining players to the lobby.

FRONT END & GUI (M)

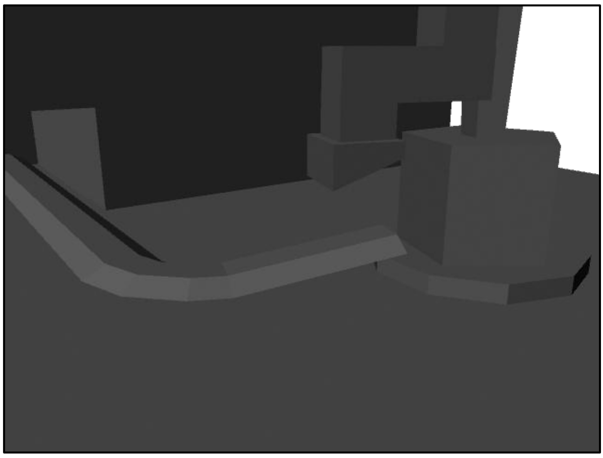
GAME & MENU FLOW (M)

This section details the flow of the menu and game states on opening the game.



The above diagram lists each of the screens/states inside of the bubbles, with the connecting arrows indicating movement between said screens. Labels along each connecting arrow indicate either button presses required, or other conditional requirements.

FRONT END VISUALS & FUNCTIONALITY (M)



The front end will be contained within a factory setting. The main view will consist of a carousel which comprises of interactive objects that allow the player to navigate the menu and start a game. Each menu will be contained on a separate side of the carousel and it will rotate to the desired screen by making selections and flicking the screen left and right.

- The objects for making menu selections will be contained in and around the carousel
- When buttons are pressed, it will play relevant animations in the background
- Interactions will trigger sounds
- Relevant background sounds will trigger to bring life to the scene
- Camera will zoom into the screen at specific points to give a fuller view of menu options

ANIMATIONS (C)

Animations for the Front End.

- Turning the carousel to each menu screen
- Steam particle effects
- Moving the camera (when showing the ball animation)
- Moving a ball down onto the conveyer belt and through the exit on the wall (when play is selected)

MENU SCREENS & MENUS (M)

This section will detail each of the menu screens, their content and functionality.

PLAY SCREEN (M)

The title menu is the first screen after the splash screens which the player will see. This screen will contain 4 buttons. These buttons will be:

- Single Player
- Multiplayer
- Party Setup
- Exit

SINGLE PLAYER MENU (C)

The Single Player screen allows the player to set up a race. This screen will allow the player to change:

- Character
- Track

It will also contain two buttons for navigation.

- Play
- Back

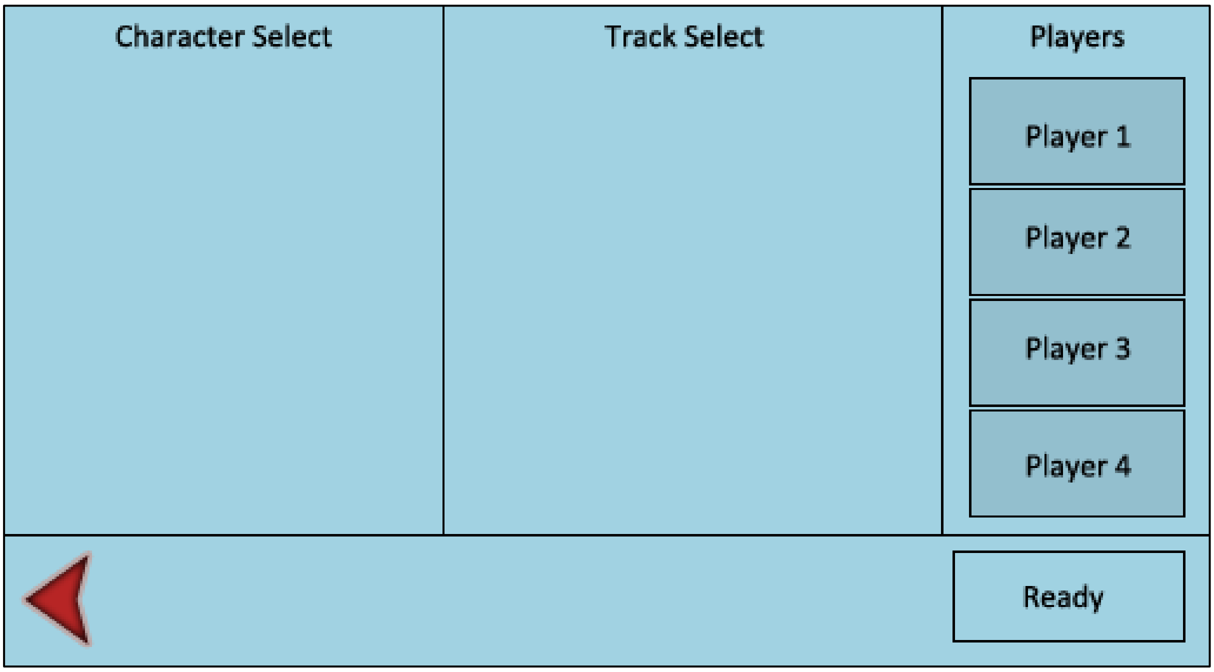
MULTIPLAYER MENU (M)

The multiplayer screen will allow each player to set up the race and their characters. It will be made up of three main panes.

- Individual player selection
- Track selection
- Player status

It will also contain two buttons:

- Ready
- Back



- The back button will play the major negative sound when pressed
- The character and track panes will require a swipe motion to change elements
- When characters switch they will play a short voice clip for that player
- Pressing the ready button will play the final confirmation sound
- Each of the player slots will show a default character silhouette until the player has readied
- If the player has readied it will show an image of the character selected

- Player 1 is in control of the track selection
- When all characters are ready, the ready button will disappear and be replaced with an on-screen three second timer
- When the three second timer has completed it will start the game

PARTY SETUP MENU (C)

The party setup screen is linked up with Facebook to allow players to invite their friends to race with them. A player can create a party of up to 4 friends.

OPTIONS SCREEN

- Volume toggle button
- Facebook button (C)
- Racing Name (text entry box) (C)

GALAXY VIEW SCREEN

- Options
- Galaxy View (C)
- Exit

PAUSE/OPTIONS (M)

The pause/options menu is opened by pressing a button located on the HUD in the game. It has three options:

- Toggle Sound
- Resume
- Quit

Some details about the menu:

- It will be a small panel under the timer at the top of the screen.
- Whilst this menu is open in a single player game, it will pause all activity in the game and game interactions.
- For a multiplayer game, it will not pause all activity and the player will still maintain the ability to turn.

OPTIONS (M)

- Volume toggle button
- Facebook button (C)
- Racing Name text entry box

PROMPTS

Opening the racing name text box will bring up a prompt. It will state that a racing name will override the use of their Facebook name.

FACEBOOK (W)

- Facebook Email
- Facebook Password
- Back

PROMPTS

- Facebook will prompt the player of how it will use his/her Facebook account.

CREDITS (M)

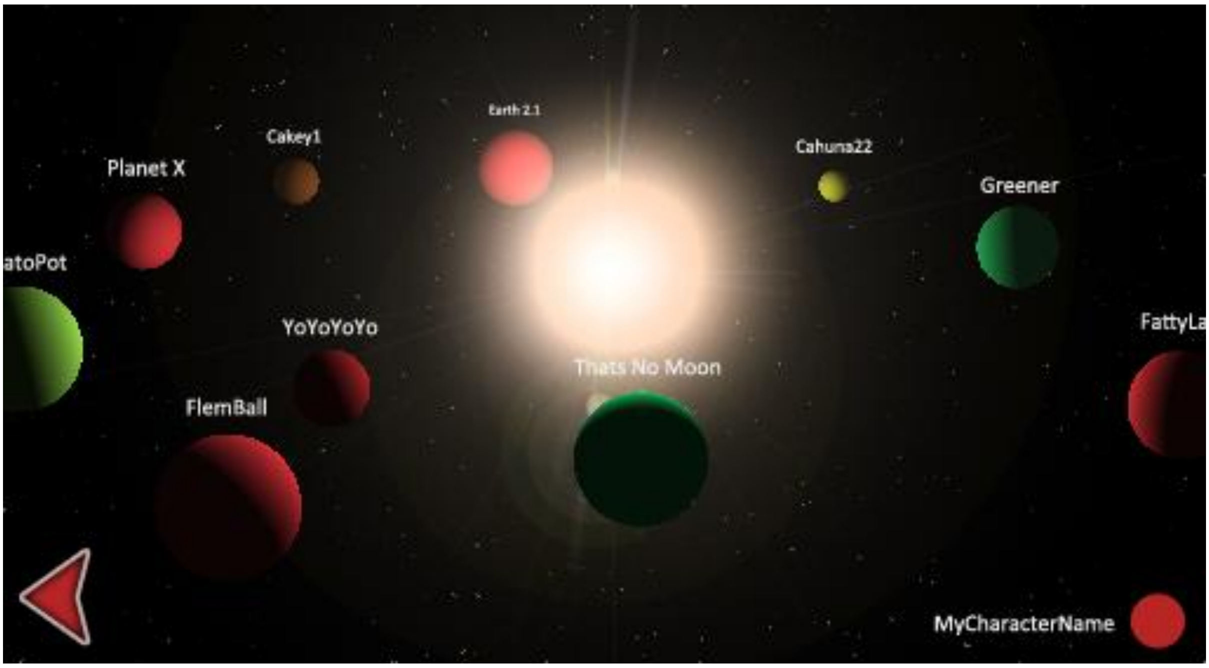
The credits screen will be a static screen with all those involved and their roles listed.

- Mustafa Çetiner
- Elinor Townsend
- William Mincher
- Ralph Norman
- Rab McQueen
- Karl Inglott

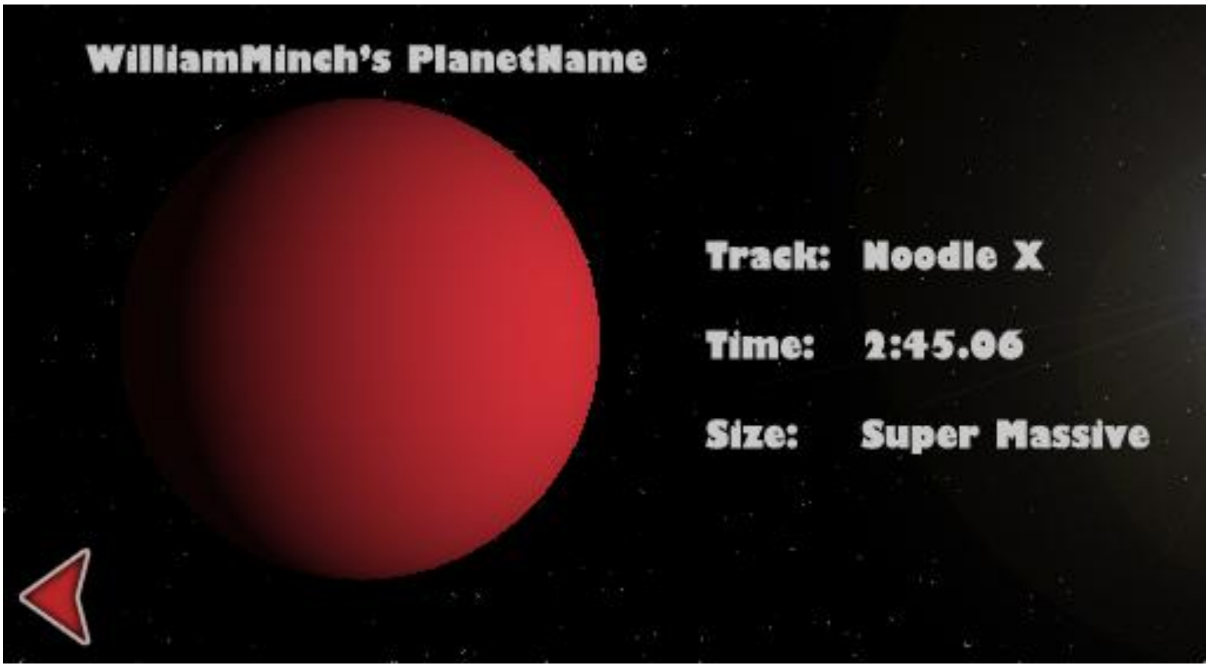
It will have a single button allowing the user to move back to the Options screen

GALAXY VIEW (C)

The galaxy view will be a place to view the leaderboards and the achievements.



In Galaxy View, each of the planets represents the highest scoring players' cake balls. The player can rotate around the sun and touch planets to get further information on the player who made it.



When viewing a planet, the player can move back to the main galaxy view by tapping the arrow button. A player can also find his/her own planets by selecting the area containing his/her username which brings up the personal panel.

GUI (M)

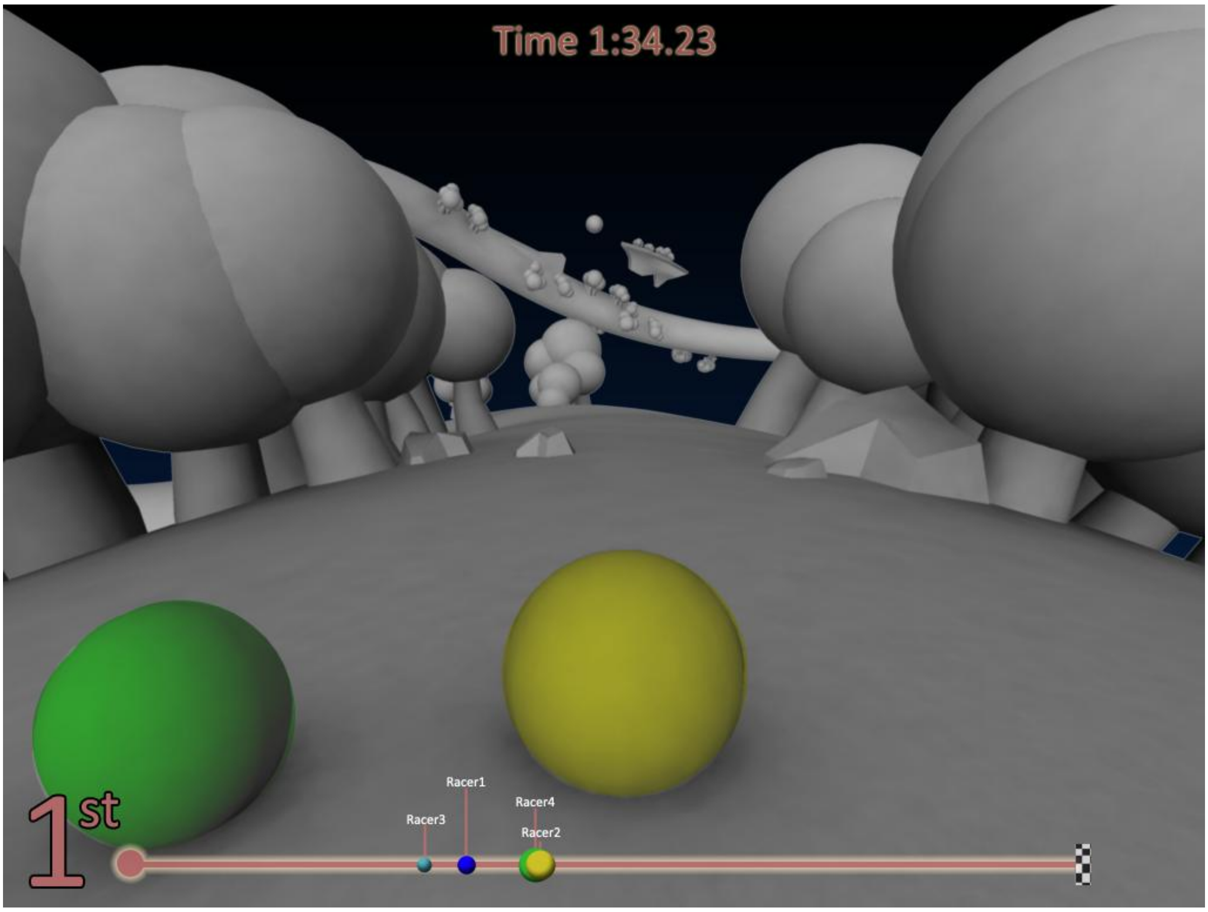
The GUI will be made up of four elements

- Options/pause button
- Time
- Position
- Race Track Line (S)

The GUI should have the following:

- The options/pause button will require a touch to open. It will show the pause/options menu.
- The position indicator will show what place the player is currently in
- The time will show how long the race has continued for.

GUI CONT…



RACE TRACK LINE (S)

The race track line is a line indicating the length of the track and each player's position in the track. It should be made up of a line, with four coloured circles representing each player. The start and end of the line should have

PAUSE/OPTIONS BUTTON

A button will be located at the top right of the screen which will open the pause/options menu. (see Pause/Options)

ACHIEVEMENTS (C)

Achievements should be implemented into the game. All of these achievements and their status can be seen in the Galaxy View. The player is notified of achievements gained when hitting the target. This could be during a race, at the end of a race or in the menus.

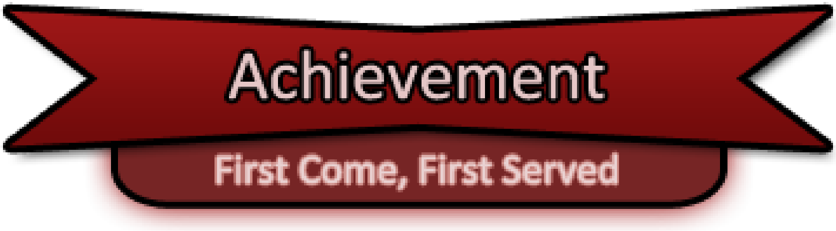
- All data on Achievements is stored locally.
- Gaining an achievement will display a prompt showing the achievement gained

Name	Description In-Game	Criteria
First come, first served		Finish in first
Ding!		Complete a race
Space balls		Play a multiplayer race
Blade Roller		Beat the AI
Mad Rockets		Watch the credits
Biscuit taker		Absorb 100 objects
Who ate all the pies?		Absorb 500 objects
Chestnut		Absorb 1000 objects
Total Éclair		Absorb 3000 objects
Cookie Monster		Absorb 10,000 objects
Jumping Jupiter!		Jump from a ramp
That's no moon!		Get on the leaderboards
		Finish a race in track 1
		Finish a race in track 2
		Finish a race in track 3
When Planets Collide		Crash into another player
		Get a time of under X:XX.XX in track 1
		Get a time of under X:XX.XX in track 2
		Get a time of under X:XX.XX in track 3
		Find the Secret Ingredient in track 1
		Find the Secret Ingredient in track 2
		Find the Secret Ingredient in track 3
On a Roll		Finish 5 races in first, consecutively
		Use a slipstream
		Finish 10 races
		Finish 50 races
		Finish 100 races
		Finish 500 races
		Finish 1000 races
		Finish 4000 races
Roll out the red carpet!		Finish 9,000 races
The cake is not a lie		Get all achievements

ACHIEVEMENT PROMPT

The achievement prompt should comprise of a box that appears from the top of the screen. The box will contain the title and description of the achievement gained.

- Achievement prompt will trigger the achievement sound
- It will appear just under the timer at the top of the screen with a bursting particle effect
- It will appear for 3 seconds before disappearing
- It will contain the Title "Achievement" and contain the title of the achievement



IN-GAME DEBUGGING

The in-game debugging should allow developers to check the background information on the game. This will better allow developers to understand bugs.

Information should be drawn onto the screen should contain the following:

- Frames Per Second
- Memory Usage
- Version Number

All of this information should be drawn on the top left side of the screen.

It should be possible to toggle the information by double tapping the area where it appears/disappears from.

ASSETS

MODELS, TEXTURES, ART & ANIMATION

This section lists the assets, file extensions and folder paths for models, textures, art and animation.

- All static models should use the .mb file extension
- All animated models should use the .fbx file extension
- All textures and art should use the .png file extension

CHARACTERS

Asset	Filename	Destination Folder Path
Character 1: Rigged & Animated Model	char1.fbx	*…Assets/Models/Characters*
Character 2: Rigged & Animated Model	char2.fbx	*…Assets/Models/Characters*
Character 3: Rigged & Animated Model	char3.fbx	*…Assets/Models/Characters*
Character 4: Rigged & Animated Model	char4.fbx	*…Assets/Models/Characters*
Character 1: Texture	char1_tex.png	*…Assets/Textures/Characters*
Character 2: Texture	char2_tex.png	*…Assets/Textures/Characters*
Character 3: Texture	char3_tex.png	*…Assets/Textures/Characters*
Character 4: Texture	char4_tex.png	*…Assets/Textures/Characters*

ANIMATIONS

Asset
Character 1 Speed 1: Running
Character 1 Speed 1: Running & Turning Left
Character 1 Speed 1: Running & Turning Right
Character 1 Speed 1: Light Collision & back to Running
Character 1 Speed 1: Heavy Collision & back to Running
Character 1 Speed 1: Running & negative gesture
Character 1 Speed 2: Running
Character 1 Speed 2: Running & Turning Left
Character 1 Speed 2: Running & Turning Right
Character 1 Speed 2: Light Collision & back to Running
Character 1 Speed 2: Heavy Collision & back to Running
Character 1 Speed 3: Running
Character 1 Speed 3: Running & Turning Left
Character 1 Speed 3: Running & Turning Right
Character 1 Speed 3: Light Collision & back to Running
Character 1 Speed 3: Heavy Collision & back to Running
Character 1 Front End & Pre-Race: Idle
Character 2 Speed 1: Running
Character 2 Speed 1: Running & Turning Left
Character 2 Speed 1: Running & Turning Right

Character 2 Speed 1: Light Collision & back to Running
Character 2 Speed 1: Heavy Collision & back to Running
Character 2 Speed 1: Running & negative gesture
Character 2 Speed 2: Running
Character 2 Speed 2: Running & Turning Left
Character 2 Speed 2: Running & Turning Right
Character 2 Speed 2: Light Collision & back to Running
Character 2 Speed 2: Heavy Collision & back to Running
Character 2 Speed 3: Running
Character 2 Speed 3: Running & Turning Left
Character 2 Speed 3: Running & Turning Right
Character 2 Speed 3: Light Collision & back to Running
Character 2 Speed 3: Heavy Collision & back to Running
Character 2 Front End & Pre-Race: Idle
Character 3 Speed 1: Running
Character 3 Speed 1: Running & Turning Left
Character 3 Speed 1: Running & Turning Right
Character 3 Speed 1: Light Collision & back to Running
Character 3 Speed 1: Heavy Collision & back to Running
Character 3 Speed 1: Running & negative gesture
Character 3 Speed 2: Running
Character 3 Speed 2: Running & Turning Left
Character 3 Speed 2: Running & Turning Right
Character 3 Speed 2: Light Collision & back to Running
Character 3 Speed 2: Heavy Collision & back to Running
Character 3 Speed 3: Running
Character 3 Speed 3: Running & Turning Left
Character 3 Speed 3: Running & Turning Right
Character 3 Speed 3: Light Collision & back to Running
Character 3 Speed 3: Heavy Collision & back to Running
Character 3 Front End & Pre-Race: Idle
Character 4 Speed 1: Running
Character 4 Speed 1: Running & Turning Left
Character 4 Speed 1: Running & Turning Right
Character 4 Speed 1: Light Collision & back to Running
Character 4 Speed 1: Heavy Collision & back to Running
Character 4 Speed 1: Running & negative gesture
Character 4 Speed 2: Running
Character 4 Speed 2: Running & Turning Left
Character 4 Speed 2: Running & Turning Right
Character 4 Speed 2: Light Collision & back to Running
Character 4 Speed 2: Heavy Collision & back to Running
Character 4 Speed 3: Running

Character 4 Speed 3: Running & Turning Left
Character 4 Speed 3: Running & Turning Right
Character 4 Speed 3: Light Collision & back to Running
Character 4 Speed 3: Heavy Collision & back to Running
Character 4 Speed 3: Running & negative gesture
Character 4 Front End & Pre-Race: Idle

ENVIRONMENT

Asset	Filename	Destination Folder Path
Tree 1: Model	Env_tree1.mb	*...Assets/Models/Environment*
Tree 2: Model	Env_tree2.mb	*...Assets/Models/Environment*
Tree 3: Model	Env_tree3.mb	*...Assets/Models/Environment*
Tree 4: Model	Env_tree4.mb	*...Assets/Models/Environment*
Tree 1: Texture	Env_tree1_tex.png	*...Assets/Textures/Environment*
Tree 2: Texture	Env_tree2_tex.png	*...Assets/Textures/Environment*
Tree 3: Texture	Env_tree3_tex.png	*...Assets/Textures/Environment*
Tree 4: Texture	Env_tree4_tex.png	*...Assets/Textures/Environment*

IMAGES AND PARTICLES

LEVEL TERRAIN

LEVEL SEGMENTS

FRONT END

IN GAME UI

SOUND EFFECTS & MUSIC

CHARACTERS

Asset	**Filename**	**Destination Folder Path**
Character 1: Positive 1	sfx_char1_pos1.aac	*...Assets/Sound/Characters*
Character 2: Positive 1	sfx_char2_pos1.aac	*...Assets/Sound/Characters*
Character 3: Positive 1	sfx_char3_pos1.aac	*...Assets/Sound/Characters*
Character 4: Positive 1	sfx_char4_pos1.aac	*...Assets/Sound/Characters*
Character 1: Positive 2	sfx_char1_pos2.aac	*...Assets/Sound/Characters*
Character 2: Positive 2	sfx_char2_pos2.aac	*...Assets/Sound/Characters*
Character 3: Positive 2	sfx_char3_pos2.aac	*...Assets/Sound/Characters*
Character 4: Positive 2	sfx_char4_pos2.aac	*...Assets/Sound/Characters*
Character 1: Positive 3	sfx_char1_pos3.aac	*...Assets/Sound/Characters*
Character 2: Positive 3	sfx_char2_pos3.aac	*...Assets/Sound/Characters*
Character 3: Positive 3	sfx_char3_pos3.aac	*...Assets/Sound/Characters*
Character 4: Positive 3	sfx_char4_pos3.aac	*...Assets/Sound/Characters*
Character 1: Negative 1	sfx_char1_neg1.aac	*...Assets/Sound/Characters*
Character 2: Negative 1	sfx_char2_neg1.aac	*...Assets/Sound/Characters*
Character 3: Negative 1	sfx_char3_neg1.aac	*...Assets/Sound/Characters*
Character 4: Negative 1	sfx_char4_neg1.aac	*...Assets/Sound/Characters*
Character 1: Negative 2	sfx_char1_neg2.aac	*...Assets/Sound/Characters*
Character 2: Negative 2	sfx_char2_neg2.aac	*...Assets/Sound/Characters*
Character 3: Negative 2	sfx_char3_neg2.aac	*...Assets/Sound/Characters*
Character 4: Negative 2	sfx_char4_neg2.aac	*...Assets/Sound/Characters*
Character 1: Negative 3	sfx_char1_neg3.aac	*...Assets/Sound/Characters*
Character 2: Negative 3	sfx_char2_neg3.aac	*...Assets/Sound/Characters*
Character 3: Negative 3	sfx_char3_neg3.aac	*...Assets/Sound/Characters*
Character 4: Negative 3	sfx_char4_neg3.aac	*...Assets/Sound/Characters*
Character 1: Race Start	sfx_char1_racestart.aac	*...Assets/Sound/Characters*
Character 2: Race Start	sfx_char2_racestart.aac	*...Assets/Sound/Characters*
Character 3: Race Start	sfx_char3_racestart.aac	*...Assets/Sound/Characters*
Character 4: Race Start	sfx_char4_racestart.aac	*...Assets/Sound/Characters*
Character 1: Race Finish	sfx_char1_racefinish.aac	*...Assets/Sound/Characters*
Character 2: Race Finish	sfx_char2_racefinish.aac	*...Assets/Sound/Characters*
Character 3: Race Finish	sfx_char3_racefinish.aac	*...Assets/Sound/Characters*
Character 4: Race Finish	sfx_char4_racefinish.aac	*...Assets/Sound/Characters*
Character 1: Flying through the air	sfx_char1_flying.aac	*...Assets/Sound/Characters*
Character 2: Flying through the air	sfx_char2_flying.aac	*...Assets/Sound/Characters*
Character 3: Flying through the air	sfx_char3_flying.aac	*...Assets/Sound/Characters*
Character 4: Flying through the air	sfx_char4_flying.aac	*...Assets/Sound/Characters*

COLLISION SOUND EFFECTS

Asset	Filename	Destination Folder Path
Ice Cream Tree	sfx_env_icecreamtree	*...Assets/Sound/Collision*
Candy Floss Tree	sfx_env_candyflosstree	*...Assets/Sound/Collision*
Player	sfx_env_player	*...Assets/Sound/Collision*
Cookie Rock	sfx_env_cookierock	*...Assets/Sound/Collision*
Strawberry	sfx_env_strawberry	*...Assets/Sound/Collision*
Coconut Mushroom	sfx_env_coconutmushroom	*...Assets/Sound/Collision*
Doughnut	sfx_env_doughnut	*...Assets/Sound/Collision*
Milkshake	sfx_env_milkshake	*...Assets/Sound/Collision*
Lollypop	sfx_env_lollypop	*...Assets/Sound/Collision*
Ice Cream Rock	sfx_env_icecreamrock	*...Assets/Sound/Collision*
Apple	sfx_env_apple	*...Assets/Sound/Collision*

UI SOUND EFFECTS

Asset	Filename	Destination Folder Path
Positive On Down	sfx_fe_posdown	*...Assets/Sound/UI*
Positive Release	sfx_fe_posrelease	*...Assets/Sound/UI*
Negative On Down	sfx_fe_negdown	*...Assets/Sound/UI*
Negative Release	sfx_fe_negrelease	*...Assets/Sound/UI*
Change Selection	sfx_fe_changesel	*...Assets/Sound/UI*
Completed	sfx_fe_complete	*...Assets/Sound/UI*
Exit	sfx_fe_exit	*...Assets/Sound/UI*

MUSIC

Asset	Filename	Destination Folder Path
Title Screen	m_title	*...Assets/Music*
Galaxy view	m_galaxy	*...Assets/Music*
Track 1	m_track1	*...Assets/Music*
Track 2	m_track2	*...Assets/Music*
Track 3	m_track3	*...Assets/Music*
Fanfare Winner	m_fanfare1	*...Assets/Music*
Fanfare Second Place	m_fanfare2	*...Assets/Music*
Fanfare Third Place	m_fanfare3	*...Assets/Music*
Fanfare Fourth Place	m_fanfare4	*...Assets/Music*

FILE NAMING CONVENTIONS

All asset names should be lower case as part of the naming convention. This includes layers/objects contained within the raw files. Filenames should contain no spaces and instead an underscore should be used.

# denotes a number used as an identifier where many objects in the same path may share the same name. This will start at 1 and increase by 1 with each object of the same type. It should contain no preceding 0's. cookie1 cookie2, cookie3, etc.

"Texture files" refers to any 2D image that is applied to a 3D model to alter the appearance. This includes normal maps, specular maps, light maps, etc.

All files should have an identifying prefix that sets assets apart from each other. This prefix will depend on the object type.

EXTERNAL FILES

External files are all of the objects created outside of Unity. This includes audio, images and models.

3D MODELS & ANIMATIONS

3D models should be appropriately named in the raw Maya file and when exported. This includes all objects contained within the Maya file (shapes, materials, rigs, etc). This allows us to easily find parts for reference within Unity if any problems should occur or for referencing in the code.

CHARACTERS

This section refers to all of the playable characters in the game.

Type	Convention	Notes
Character fbx files	char#	It is assumed we will be using a single fbx files for each animated character.
Character Model Parts (eg. Head, left leg, right leg, rig)	char#_bodypart	
Character Texture files	char#_tex	tex should be replaced depending on the type of texture: spec, norm, disp, etc.

ENVIRONMENT OBJECTS

This section refers to all objects contained within the levels (with the exception of the level terrain).

Type	Convention	
fbx/ma/mb files	env_type#	The type refers to the object type, so env_marshmallow1, env_doughnut13 etc.
Model Parts (eg. Trunk, canopy, doughnut)	env_type#_part	
Texture files	env_type#_tex	tex should be replaced depending on the type of texture: spec, norm, disp, etc.

FRONT END

This section refers to all objects that will make up the front end. It encompasses Models and Art & Images.

MODELS

Type	Convention	
fbx/ma/mb files	fe_type#	The type refers to the object type, so fe_planet4, fe_screen etc.
Model Parts (eg. Trunk, canopy, doughnut)	fe_type#_part	
Texture files	fe_type#_tex	tex should be replaced depending on the type of texture: spec, norm, disp, etc.

ART & IMAGES

AUDIO

SOUNDS & MUSIC

Type	Convention	
Tactile response sound effects	sfx_function#	These are the sounds that make up the button presses in the game
In-game feedback sound effects	sfx_type#	These are the sounds that give feedback to in-game interactivity including collision sounds, moving sounds etc. *type* refers to the type as listed in the asset list.
Character sounds	sfx_char#_type#	The char# refers to the character which the sound has been created for. The type prefix refers to the type of sound as listed in the asset list.
Music	m_usage#	The usage suffix refers to the usage in the game, such as m_title, m_ingame1, m_ingame2 etc.

INTERNAL FILES

Internal Files are the objects created in Unity from external objects. This includes prefabs, scripts and materials.