Snack Attack Technical Design Document

17.07.2016

Author: David Azouz CIT196497
https://www.facebook.com/SnackAttackGame/

Academy of Interactive Entertainment
GAM100-A C01
October 23rd, 2016
Instructor: Toan Huynh

Table of Contents

Team Boss Fighters

Project Scope

August 3rd, 2016

Project Overview

1. Game Concept and Description

Snack Attack is a local multiplayer 3D brawler where up to four-players battle it out to get as many kills/ defeats as possible before the time is up. The game has a stylistic art style to match its PG rating which seeks inspiration from games like Power Stone, Gang Beasts, and Smash Bros. Cloudy with a Chance of Meatballs and Plants vs. Zombies are also an inspiration for the art style and direction.

Players choose a snack to play as and must attack other players for their "blobs" to drop - to then consume and grow. As they grow, their appearance will change based on how powerful they are. Players fight for dominance of the arena to become "the Boss" for increased damage effect, however they are now the biggest target on the field!

2. System Requirements

The project will aim for "Joystick Only" controls with partial keyboard and mouse support.

PC PLATFORM

MINIMUM:

- OS: Windows 7 64-bit
- Processor: Intel Core i3-4160 @ 2.60GHz
- Memory: 3 GB RAM
- Graphics: NVIDIA® GeForce® GTX 480 or better
- DirectX: Version 11
- Hard Drive: 4GB available space
- Sound Card: DirectX compatible soundcard or onboard chipset

RECOMMENDED:

- OS: Windows 7 64-bit and above
- Processor: Intel Core i5-4460 @3.20GHz or AMD FX-9370
- Memory: 4 GB RAM
- Graphics: NVIDIA® GeForce® GTX 760 or AMD Radeon R7 370
- DirectX: Version 11
- Hard Drive: 4 GB available space
- Sound Card: DirectX compatible soundcard or onboard chipset

XBOX ONE PLATFORM

- Console: Xbox One 500GB Console or greater
- Hard Drive: 3GB available space
- HD Video Output: 720p,1080i,1080p

PLAYSTATION 4 PLATFORM

- Console: Playstation 4 500GB Console or greater
- Hard Drive: 3GB available space
- HD Video Output: 720p,1080i,1080p

3. Implementation Plan

Meetings shall take place every week on a Tuesday. Members will generally have a week to complete designated tasks. After every main stage of the project i.e. Alpha, Beta, Gold, completion there shall be a retrospective meeting to discuss what went wrong and what can be done better.

4. Class List With Class Details

Scripts:

For Prototype we will spawn the same set of four characters each time. A character select screen will be introduced during Alpha and shall be completed by the end of Beta.

Prototype/ First Playable
- GameManager - PlayerManager: spawns a set amount of players - PlayerController - SpawnManager - Combat: Melee - BossBlobs: what characters drop when hit and what other players must collect. - Camera - U.I.: health/ power
Alpha/ Feature Lock
- Snack Brain system (all players will have these values)   - Base Class: each base class i.e. RockyRoad, Princess Cake, will have these values i.e. same Meshes, animations   - Sub Class: each unique class will have these properties. I.e. different Materials (skins) - U.I.:   - Menu: Player Select, Level Select, Timer Select   - In-Game: Score (kills), Scoreboard - PlayerManager: creates players based on choices - Combat: Melee, Shield

Beta/ Asset Lock
- Combat: - Animation feedback   - Attack commit   - Effects implementation e.g.:   - Emissive map(s) when hit and for fists - U.I.: multiple controller functionality - SoundManager (Audio) - Effect - Environmental pickups/ weapons
Gold
- Combat: - AoE attacks   - Shockwave shader   - Collision boxes - Audio: correct sounds hooked up - Code Refactor - Podium showcasing winners

2. Art/ models

For the Prototype stage:

- Vertex Colors will be used for assets to speed development. At a later stage assets will be textured.
- Scale will determine evolution.

Weak versions (both RR and PC) will share the same Light and Heavy attacks.

Prototype/ First Playable
- Art Bible - Concept Art - Character Models: Neut: Rocky Road, Princess Cake - Character Animations: - Idle   - Walk   - Jump   - Light Attack   - Heavy Attack   - Block - Character blobs: icecream, cake - Environmental Assets - Team/ game logos

Alpha/ Feature Lock

- Begin modeling different states for both classes: weak and boss
- Trail Renders
- AoE Attacks
- Emissive map for fists
- Environmental Assets:
    - Kitchen
    - Banquet
- U.I. art

Beta/ Asset Lock

- Textures added for Character Models
- Re-texture and unwrap environmental assets
- Effects implementation
    - Jump
    - Combat:
        - Trail Renders for Light and Heavy Attacks
        - Emissive map for fists
        - AoE Attacks for Heavy
        - Unique effects for Boss attacks
    - Block
    - Transformation/ Evolution
    - Death
    - Emissive map when hit
- Lighting
- Polish models
- Refine weight painting

Gold

- Character Animations:
    - Boss Heavy Attack/ AoE unique for:
        - Rocky Road
        - Princess Cake
    - Neutral Light Attack, unique for:
        - Rocky Road
        - Princess Cake
- Audio: correct sounds hooked up
- Code Refactor
- Podium showcasing winners

5. Class Interaction and Interfaces

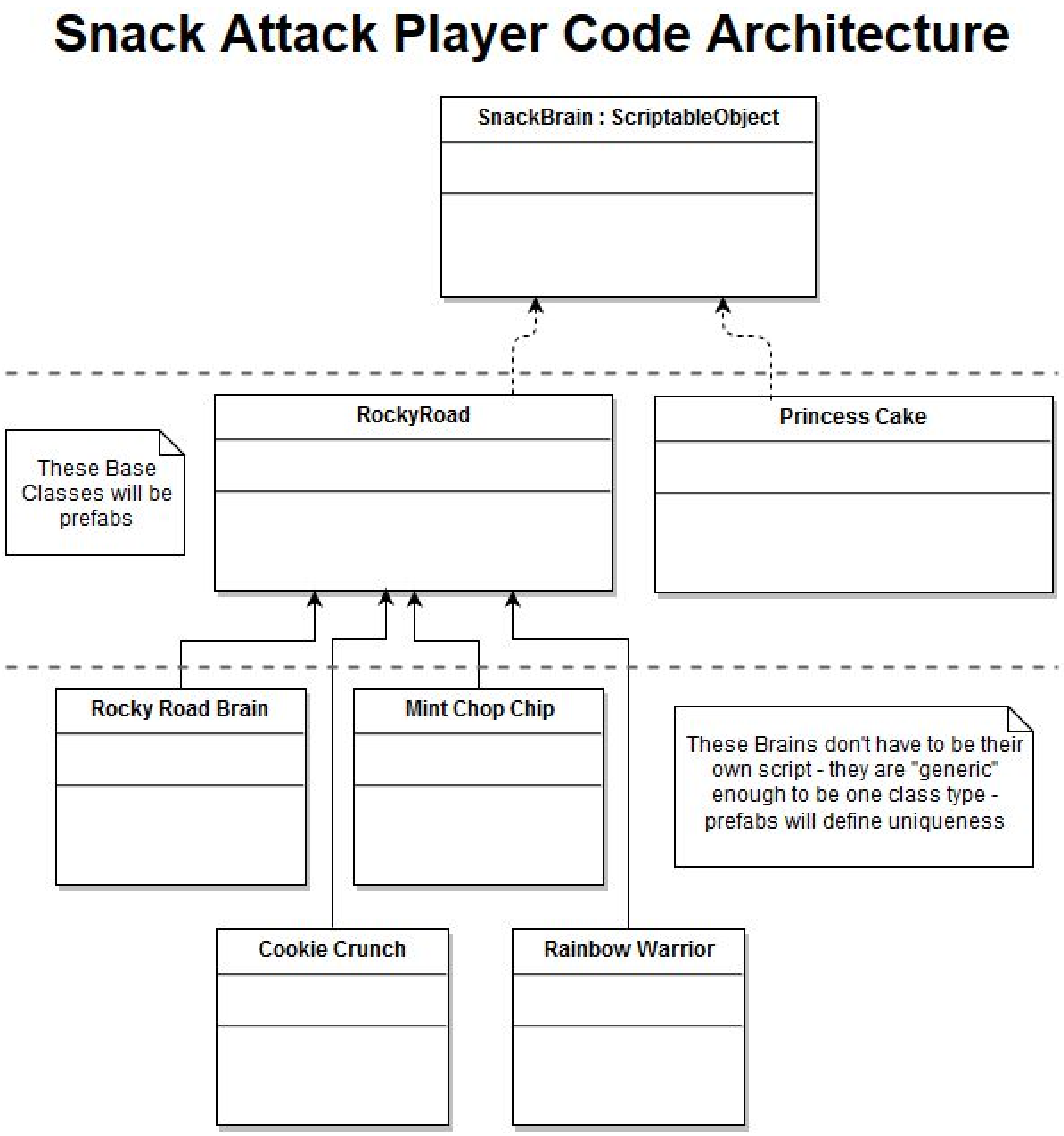

Snack Attack Player Code Architecture
SnackBrain : ScriptableObject
RockyRoad
Princess Cake
These Base Classes will be prefabs
Rocky Road Brain
Mint Chop Chip
These Brains don't have to be their own script - they are "generic" enough to be one class type - prefabs will define uniqueness
Cookie Crunch
Rainbow Warrior

Figure 1.2 - UML for how characters will be created and stored.

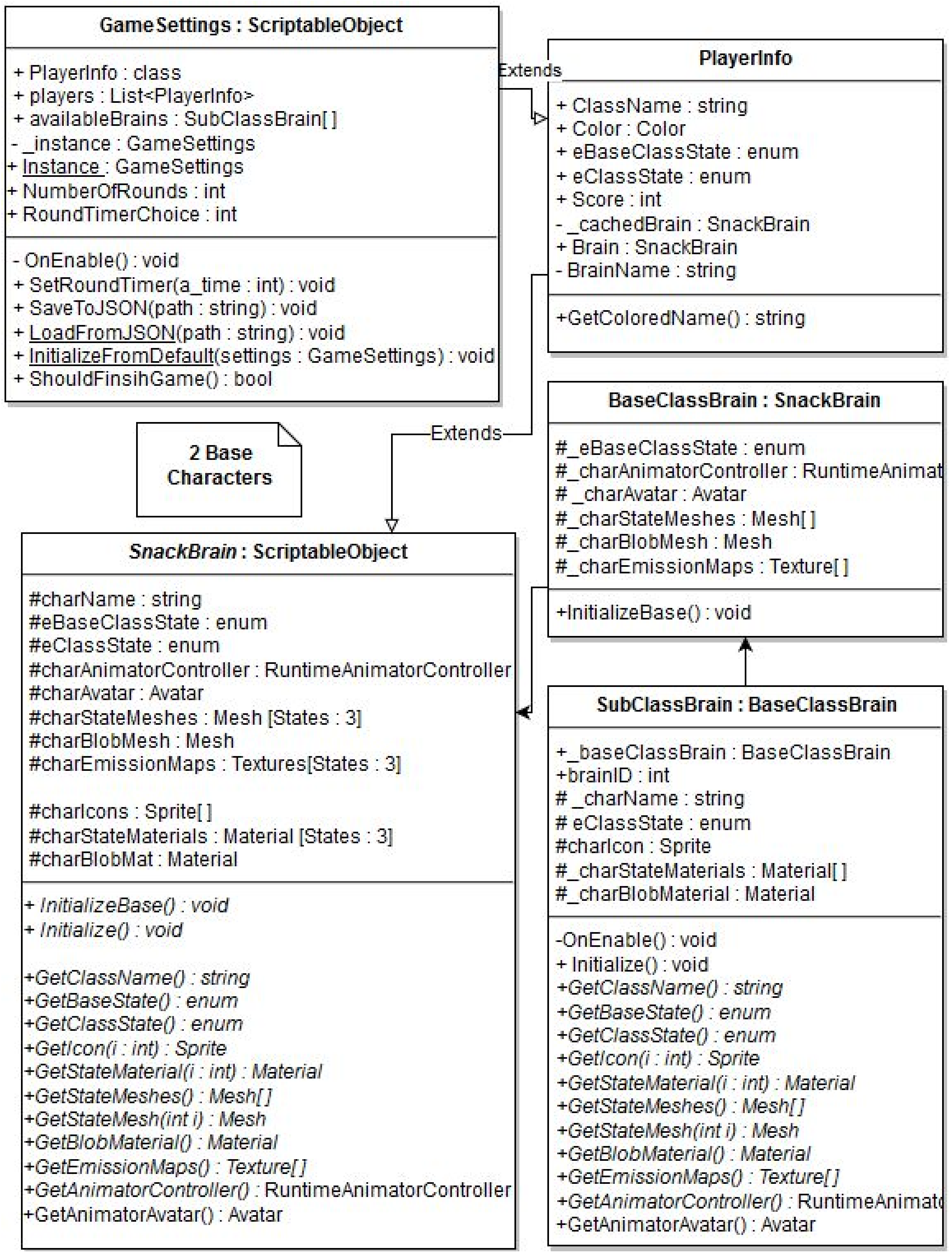

GameSettings : ScriptableObject
+ PlayerInfo : class
+ players : List<PlayerInfo>
+ availableBrains : SubClassBrain[ ]
- _instance : GameSettings
+ Instance : GameSettings
+ NumberOfRounds : int
+ RoundTimerChoice : int
- OnEnable() : void
+ SetRoundTimer(a_time : int) : void
+ SaveToJSON(path : string) : void
+ LoadFromJSON(path : string) : void
+ InitializeFromDefault(settings : GameSettings) : void
+ ShouldFinsihGame() : bool
Extends
PlayerInfo
+ ClassName : string
+ Color : Color
+ eBaseClassState : enum
+ eClassState : enum
+ Score : int
- _cachedBrain : SnackBrain
+ Brain : SnackBrain
- BrainName : string
+GetColoredName() : string
2 Base Characters
Extends
BaseClassBrain : SnackBrain
#_eBaseClassState : enum
#_charAnimatorController : RuntimeAnimat
# _charAvatar : Avatar
#_charStateMeshes : Mesh[ ]
#_charBlobMesh : Mesh
#_charEmissionMaps : Texture[ ]
+InitializeBase() : void
SnackBrain : ScriptableObject
#charName : string
#eBaseClassState : enum
#eClassState : enum
#charAnimatorController : RuntimeAnimatorController
#charAvatar : Avatar
#charStateMeshes : Mesh [States : 3]
#charBlobMesh : Mesh
#charEmissionMaps : Textures[States : 3]
#charIcons : Sprite[ ]
#charStateMaterials : Material [States : 3]
#charBlobMat : Material
+ InitializeBase() : void
+ Initialize() : void
+GetClassName() : string
+GetBaseState() : enum
+GetClassState() : enum
+GetIcon(i : int) : Sprite
+GetStateMaterial(i : int) : Material
+GetStateMeshes() : Mesh[ ]
+GetStateMesh(int i) : Mesh
+GetBlobMaterial() : Material
+GetEmissionMaps() : Texture[ ]
+GetAnimatorController() : RuntimeAnimatorController
+GetAnimatorAvatar() : Avatar
SubClassBrain : BaseClassBrain
+_baseClassBrain : BaseClassBrain
+brainID : int
# _charName : string
# eClassState : enum
#charIcon : Sprite
#_charStateMaterials : Material[ ]
#_charBlobMaterial : Material
-OnEnable() : void
+ Initialize() : void
+GetClassName() : string
+GetBaseState() : enum
+GetClassState() : enum
+GetIcon(i : int) : Sprite
+GetStateMaterial(i : int) : Material
+GetStateMeshes() : Mesh[ ]
+GetStateMesh(int i) : Mesh
+GetBlobMaterial() : Material
+GetEmissionMaps() : Texture[ ]
+GetAnimatorController() : RuntimeAnimat
+GetAnimatorAvatar() : Avatar

Figure 1.3- “Snack Brain” UML diagram of code architecture.

6. Player and Non-Player Character systems and behaviours

Player: - Characters will navigate a 3D environment on all axes - Player will have melee combat functionality	**Camera:** - The Camera will track all four players and zoom in and out based on their positions
U.I.: In Game: - A ring on each player representing their level of evolution (blobs) - U.I. specifies how many kills a player has gotten - Scoreboard tracks kills and deaths Menu: - Main Menu that contains "Start", "Controls", "Credits", and "Quit". - A Player Selection screen will be available - In addition to a level select & - A timer select	**SnackBrain:** - This along with BaseClassBrain and SubClassBrain work to allow for different configurations of characters that share common properties. - BaseClassBrain defines: - Animations - Meshes - SubClassBrain: - Materials (skins) - Icon -
Game Manager: - Game loop happens here.	**Player Manager:** - Handles creation of characters based off players selections with the help of the brain system.

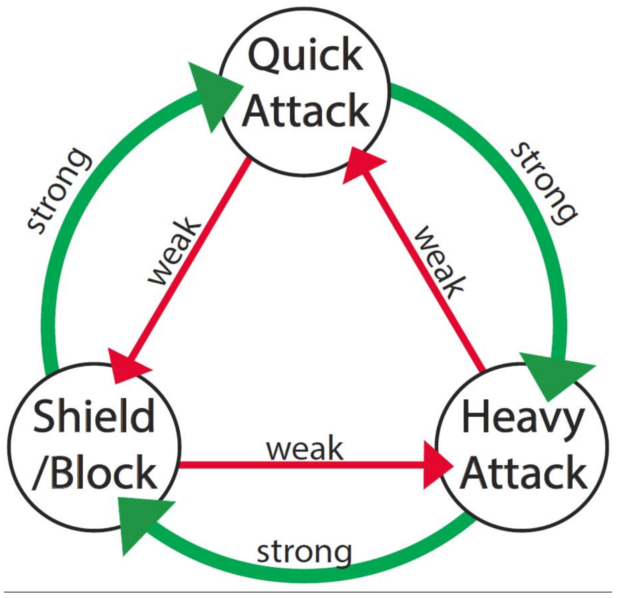


Figure 1.4- Golden Triangle of *Snack Attack's* combat system

7. Game State Diagrams

User Interface:

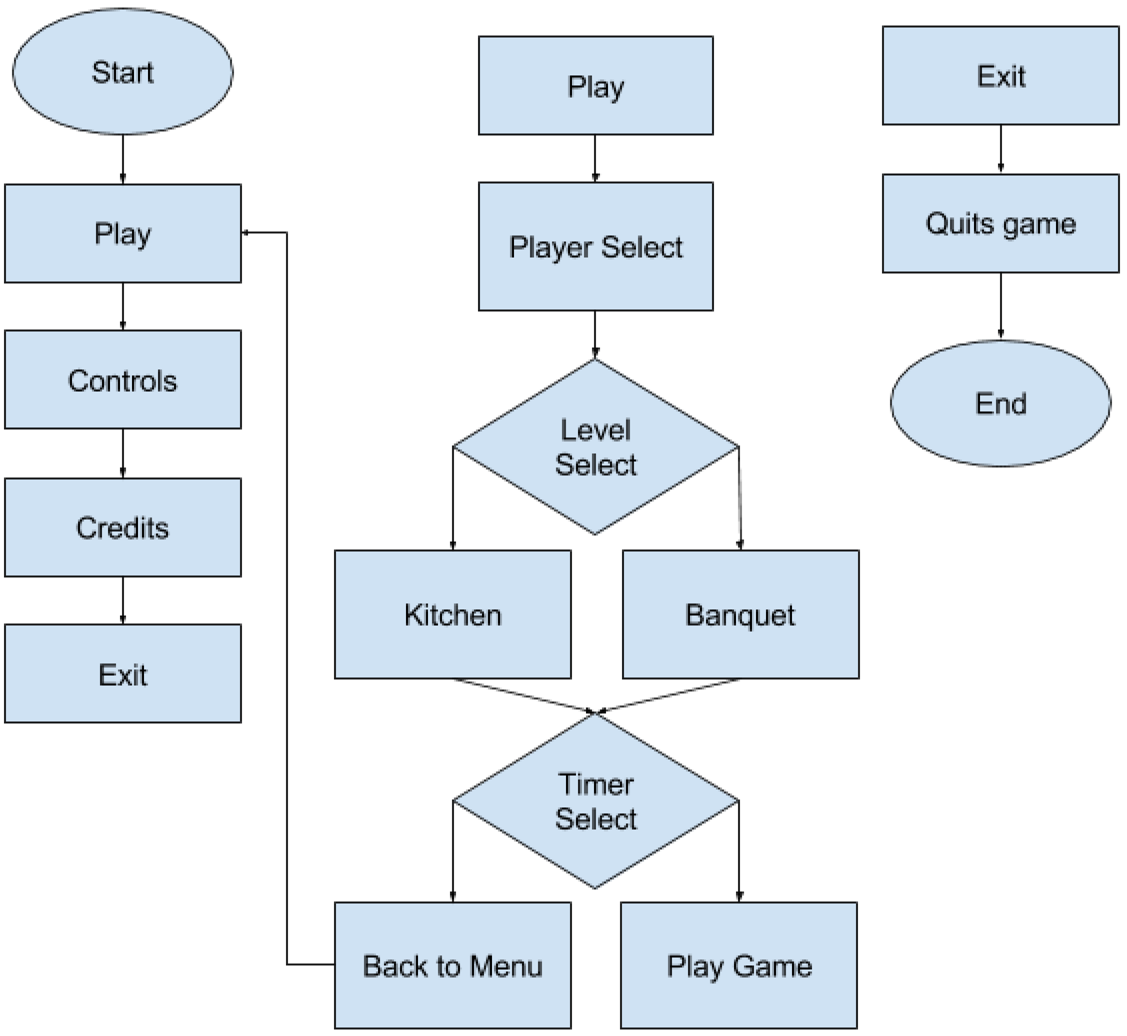


Figure 1.5- Snack Attack U.I. Flowchart

Title Screen:

Presents the game’s title and loads any assets (Game Manager) that will remain throughout play.

Main Menu:

The start-up menu will consist of a 2D menu with the following choices:

- "Start" – This opens the Player Select Menu.
- "Controls" – This opens the Controls Menu, specifying button layouts.
- "Credits" – This opens the Credits Menu, revealing who worked on the game.
- "Quit" – Quits the game and returns to the Windows desktop/ unplays the game in editor.

A particle effect will replicate food/ snacks falling in the background.

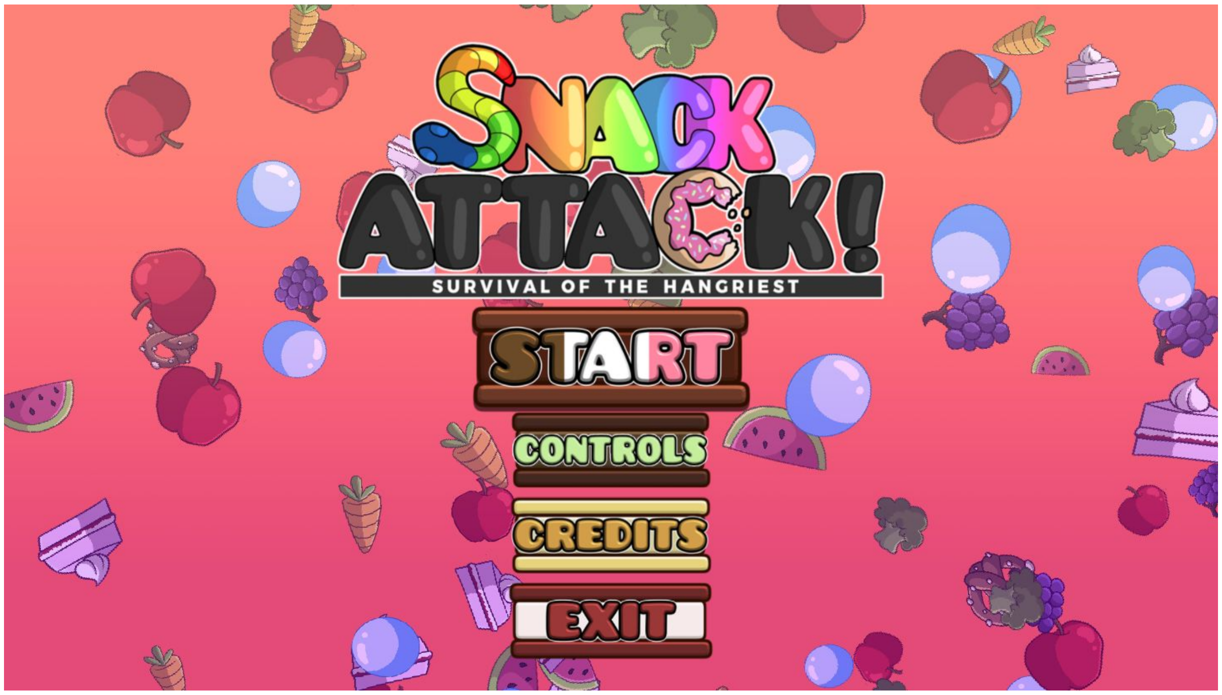
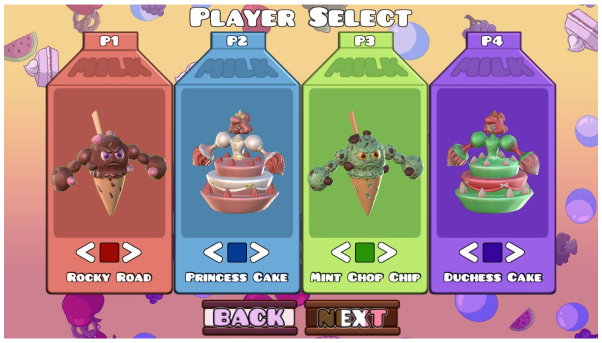
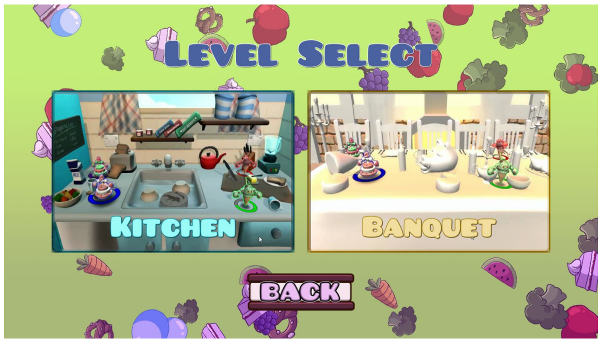
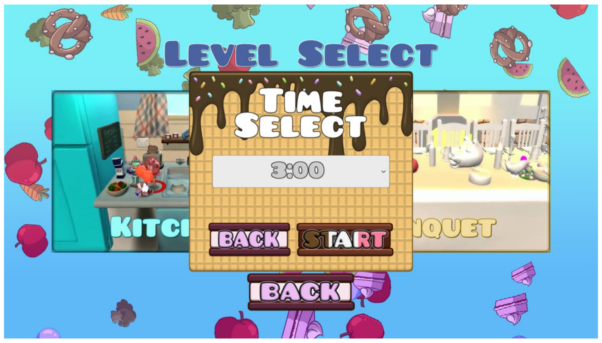


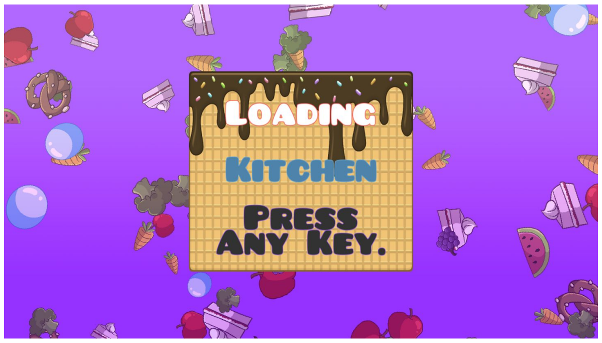
Figure 1.6- *Snack Attack* Main Menu

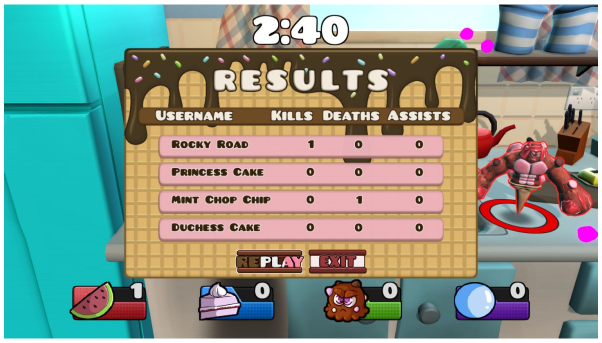
Figures 1.7-13 below: Player Select, Level Select, Timer Select, and a loading bar











Username	Kills	Deaths	Assists
Rocky Road	1	0	0
Princess Cake	0	0	0
Mint Chop Chip	0	1	0
Duchess Cake	0	0	0

<u>Player Selection:</u>

Animated 3D models give life to characters as players choose a character to play as in the game. Analog sticks will be used to cycle through playable characters animating the directed players' arrow upon cycle.

<u>Level Selection:</u>

Video footage of each level will showcase the environments for players to battle it out in.

<u>Timer Selection:</u>

"1:00", "3:00" or "5:00" will be the options for how long players can play for in a round.

<u>Controls:</u>

What buttons to push to give the desired outcome.

<u>Credits:</u>

Outlines who worked on the game and what department, whether it be: art, programming, or design.

<u>Quit:</u>

Quits the game in a build and unplays the game in editor.

<u>Scoreboard:</u>

Displays kills and deaths of each player.

<u>Times Up!:</u> announces when the time is up. Then presents the scoreboard.

8. Game State

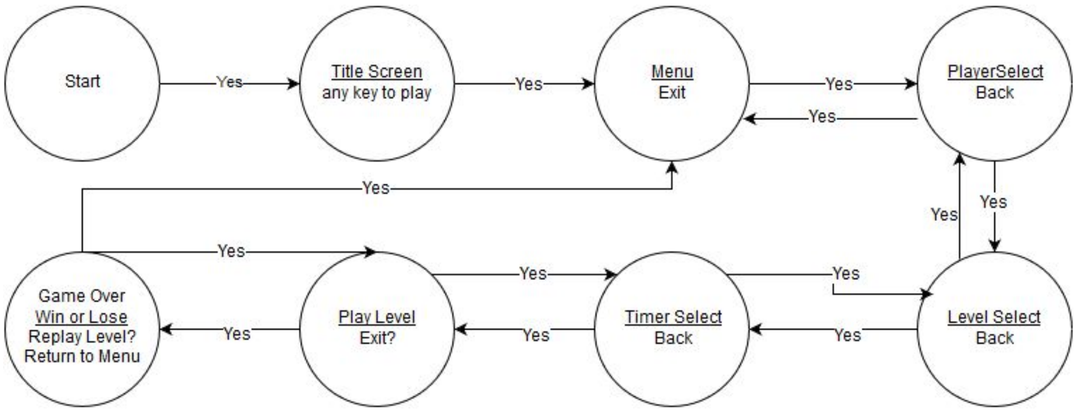


Figure 1.14- Game State Diagram

Gameplay

9. Game Controls and Camera

Players will navigate a 3D environment with a 3rd-Person view. The camera's perspective is from a "helicopter view". The camera views all players and zooms in and out based on their positions.

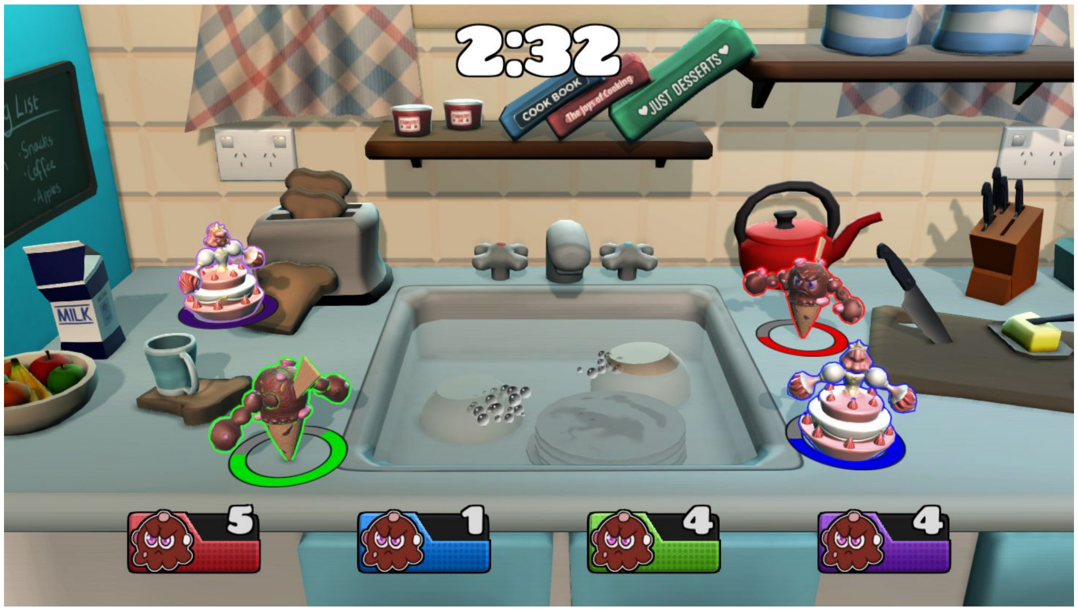


Figure 2.1- Early capture of "Snack Attack" and position of camera.

PC and Xbox One:

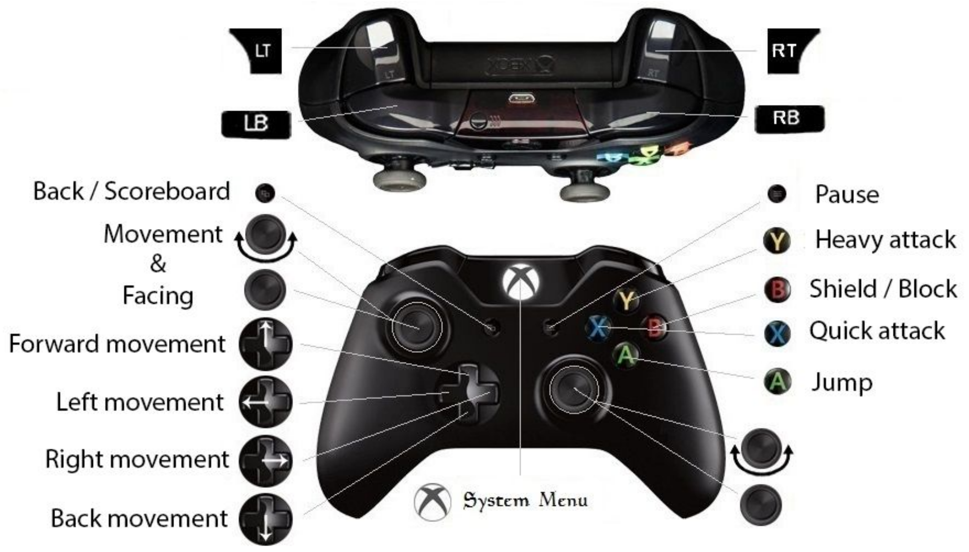


Figure 2.2- Joystick Controls for Snack Attack

Code Overview

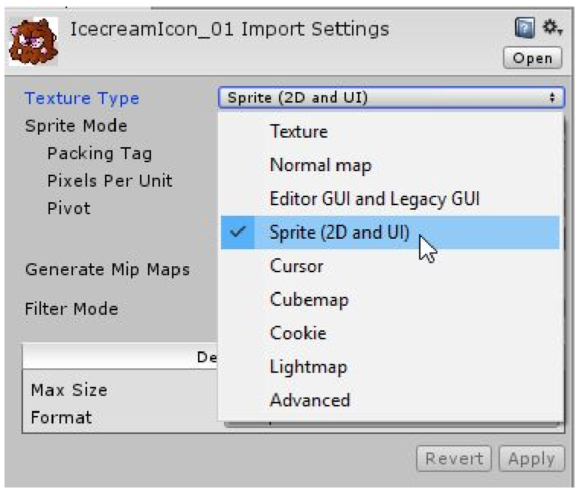
1. File Formats

Code – Unity using C#. All code should be contained in standard C# source files (.cs). The project workspace will be contained within a Microsoft Visual Studio .NET solution (.sln). Levels (Scenes) will be contained within "Unity Scene File" files (.unity).

Art - Please refer to the graphics and audio sections for their specific file format requirements.

- .FBX (Animations)
- .png (transparency) - mainly used for textures and 2D sprites.
- .shader

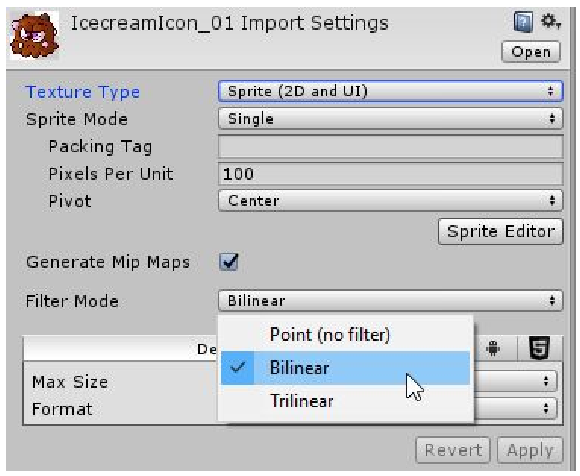
2D Art Assets: will be saved out as .png files for use in unity. ".psd" files will be stored on Google Drive for everyone to access and edit if necessary.
Within Unity, the **Texture Type** must be set to "Sprite (2D and UI). **Filter Mode** may be set to "Bilinear" but can remain "Point Filter" for higher quality. Press "Apply".



IcecreamIcon_01 Import Settings

Texture Type: Sprite (2D and UI)

Figure 2.3



IcecreamIcon_01 Import Settings

Texture Type: Sprite (2D and UI)
Sprite Mode: Single
Packing Tag:
Pixels Per Unit: 100
Pivot: Center
Generate Mip Maps: ✓
Filter Mode: Bilinear

Figure 2.4

Audio - ".WAV" will be the prefered audio format and ".mp3"'s will be used as a last resort.
3D Sound preferred.

2. Programs/ Software to be used

- Unity3D 5.4
- SourceTree
- GitHub
- Google Docs, Sheets etc.
- ZBrush
- Autodesk Maya
- 3D-Coat

- Adobe Photoshop CC
- xNormal

Plugins:

- BonesPro for Autodesk 3ds Max
- InControl (to be considered)

3. File Budgets/ Restrictions

Art Budgets:

- Textures 1024x1024
- U.I. Sprites 512x512
- 2000 MAX polygon count
- *Vertex Coloured Painted Lit TODO: what's it called?

For First Playable/ Alpha, assets are to be Vertex Coloured to speed up the art process.

Beta is where assets will be textured to improve quality and lighting functionality.

4. Naming Conventions

Programming:

```
////////////////////////////////////////////////////////
//      File Name:      (File Name)
//      Author:         FirstName LastName
//      Date Created:   (Date)
//      Brief:          Description of code below
////////////////////////////////////////////////////////

class SampleClass
{
    /// <summary>
    /// use the m_ standard, with camel case.
    /// (Lower case start for variable, upper
    /// case start for classes/ structs and functions)
    /// </summary>

    //initialisations on separate lines
    int m_sampleInt;
    int m_sampleInt2;

    bool m_sampleBool;

    int SampleFunc(int myVar)
    {
        //inline if/ else are accepted (same for functions)
        if (!m_sampleBool) return 0;
        else return myVar;
    }

    //ternary operators also okay, within reason.
    int SampleFunc2(int myVar)
    {
        return !m_sampleBool ? myVar : 0;
    }
}
```

Figure 2.5 - Coding conventions

Edit arguments must have the “a_" prefix.

Single line if statements are not okay.

Multi-Word names should have each word's first letter be capitalized.

Variables with more than one word should be camel case e.g. sampleInt.

Variables should use standard Hungarian notation whenever possible. This notation should always come before the variable's actual name and should be entirely lowercase. E.g. iMax, this variable indicates a type of integer.

Functions names need only adhere to the Multi-Word naming conventions, parameters should be in Hungarian notation with their purpose defined in the name.

Art:

Asset Names:

Environment Assets: AssetName_(#)_Type
Characters: CharacterName_Stage_Type
Blobs: CharacterName_Blob_Type

Sprites: SpriteName_SpriteSheet
Particles: ParticleName_PE
Animations:
RigName_AnimationName_Anim

Texture Types:

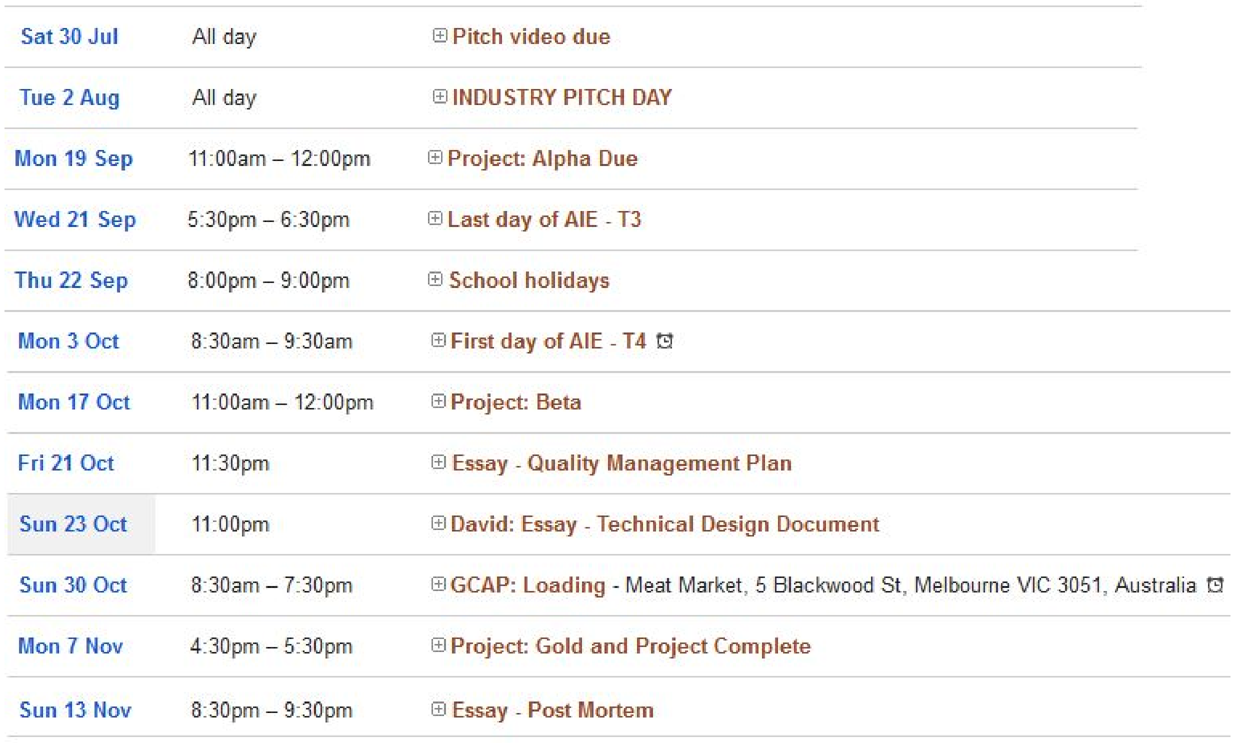
Ambient Occlusion: AO
Normal: Nrml
Specular: Spec

Diffuse: Diff
Geometry: Geo
Materials: Mat
Emissive: Emis

Character Stages:

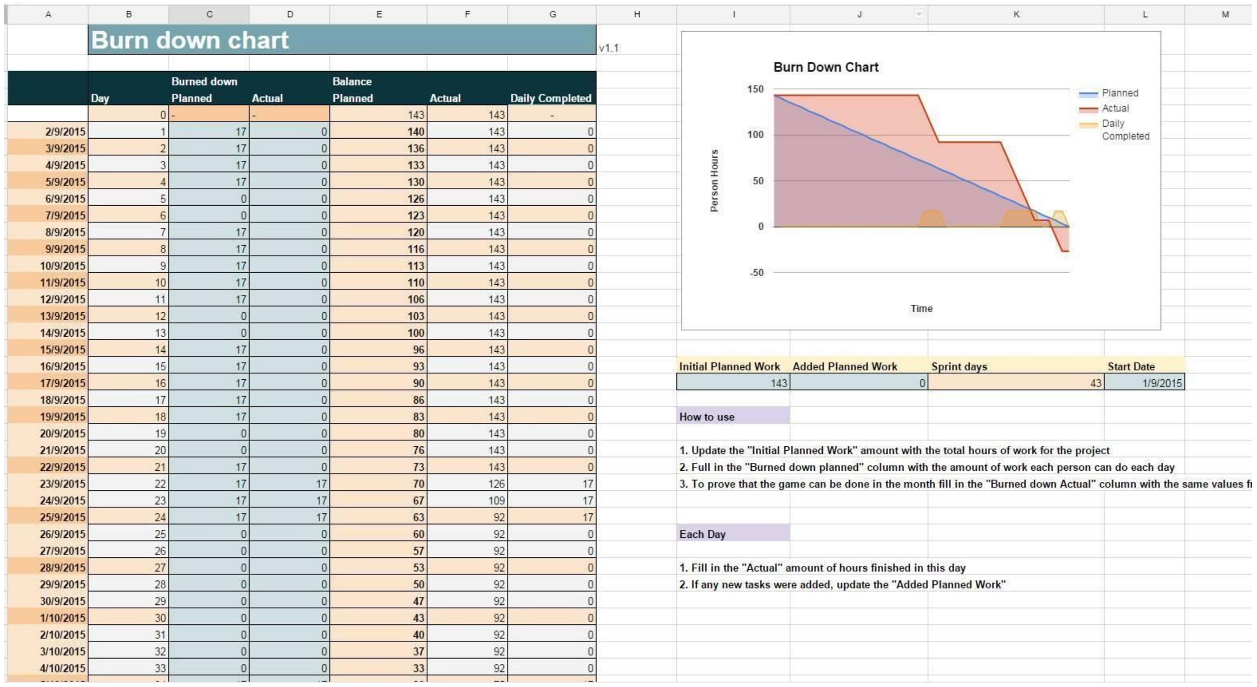
Stage 1: Weak
Stage 2: Neut
Stage 3: Boss

High-Level Timeline/ Schedule and Milestones for project completion



Sat 30 Jul	All day	⊞ Pitch video due
Tue 2 Aug	All day	⊞ INDUSTRY PITCH DAY
Mon 19 Sep	11:00am – 12:00pm	⊞ Project: Alpha Due
Wed 21 Sep	5:30pm – 6:30pm	⊞ Last day of AIE - T3
Thu 22 Sep	8:00pm – 9:00pm	⊞ School holidays
Mon 3 Oct	8:30am – 9:30am	⊞ First day of AIE - T4 ⏰
Mon 17 Oct	11:00am – 12:00pm	⊞ Project: Beta
Fri 21 Oct	11:30pm	⊞ Essay - Quality Management Plan
Sun 23 Oct	11:00pm	⊞ David: Essay - Technical Design Document
Sun 30 Oct	8:30am – 7:30pm	⊞ GCAP: Loading - Meat Market, 5 Blackwood St, Melbourne VIC 3051, Australia ⏰
Mon 7 Nov	4:30pm – 5:30pm	⊞ Project: Gold and Project Complete
Sun 13 Nov	8:30pm – 9:30pm	⊞ Essay - Post Mortem

Figure 3.1 - Calendar for milestones



Burn down chart v1.1

		Burned down		Balance		
	Day	Planned	Actual	Planned	Actual	Daily Completed
	0	-	-	143	143	-
2/9/2015	1	17	0	140	143	0
3/9/2015	2	17	0	136	143	0
4/9/2015	3	17	0	133	143	0
5/9/2015	4	17	0	130	143	0
6/9/2015	5	0	0	126	143	0
7/9/2015	6	0	0	123	143	0
8/9/2015	7	17	0	120	143	0
9/9/2015	8	17	0	116	143	0
10/9/2015	9	17	0	113	143	0
11/9/2015	10	17	0	110	143	0
12/9/2015	11	17	0	106	143	0
13/9/2015	12	0	0	103	143	0
14/9/2015	13	0	0	100	143	0
15/9/2015	14	17	0	96	143	0
16/9/2015	15	17	0	93	143	0
17/9/2015	16	17	0	90	143	0
18/9/2015	17	17	0	86	143	0
19/9/2015	18	17	0	83	143	0
20/9/2015	19	0	0	80	143	0
21/9/2015	20	0	0	76	143	0
22/9/2015	21	17	0	73	143	0
23/9/2015	22	17	17	70	126	17
24/9/2015	23	17	17	67	109	17
25/9/2015	24	17	17	63	92	17
26/9/2015	25	0	0	60	92	0
27/9/2015	26	0	0	57	92	0
28/9/2015	27	0	0	53	92	0
29/9/2015	28	0	0	50	92	0
30/9/2015	29	0	0	47	92	0
1/10/2015	30	0	0	43	92	0
2/10/2015	31	0	0	40	92	0
3/10/2015	32	0	0	37	92	0
4/10/2015	33	0	0	33	92	0

Initial Planned Work	Added Planned Work	Sprint days	Start Date
143	0	43	1/9/2015

How to use

1. Update the "Initial Planned Work" amount with the total hours of work for the project
2. Full in the "Burned down planned" column with the amount of work each person can do each day
3. To prove that the game can be done in the month fill in the "Burned down Actual" column with the same values fı

Each Day

1. Fill in the "Actual" amount of hours finished in this day
2. If any new tasks were added, update the "Added Planned Work"

Figure 3.2 - Last year's burn down chart as I’m still waiting after two months for this years Burn down chart.

Approval and Authority to Proceed

We approve the project as described above, and authorise the team to proceed.

Name	Title	Date
Toan Huynh	Project Sponsor	23/10/2016
Adam McHenry	Project Manager	23/10/2016
Duncan Wilson	Producer	23/10/2016

	23/10		23/10
Approved By	Date	Approved By	Date

References

Gamma et al, 2004, *Narbacular Drop Technical Design Document*, DigiPen, USA. https://www.digipen.edu/fileadmin/website_data/gallery/game_websites/NarbacularDrop/documents/narbacular_drop_technical_design_document.pdf.

Gang Beats on Steam http://store.steampowered.com/app/285900/

Kirill Fakhroutdinov. 2016. *UML class is a classifier which describes a set of objects that share the same features, constraints, semantics (meaning).*. [ONLINE] Available at: http://www.uml-diagrams.org/class.html. [Accessed 22 October 2016].

https://store.playstation.com/#!/en-us/games/lastfight/cid=UP1066-CUSA04447_00-LASTFIGHTRICHARD

LASTFIGHT on Steam. 2016. LASTFIGHT on Steam. [ONLINE] Available at: http://store.steampowered.com/app/443450/. [Accessed 06 October 2016].

Piranaking, 2016, *LASTFIGHT*, PC, PS4, XBO, Paris, France. http://lastfightgame.com/en/.

UML, Abstract Classes and Methods, and Interfaces | The Oxford Math Center. 2016. *UML, Abstract Classes and Methods, and Interfaces | The Oxford Math Center*. [ONLINE] Available at: http://www.oxfordmathcenter.com/drupal7/node/35. [Accessed 22 October 2016].

Unity Technologies. 2016. *Unity - Manual: Art Asset Best Practice Guide*. [ONLINE] Available at: http://docs.unity3d.com/Manual/HOWTO-ArtAssetBestPracticeGuide.html. [Accessed 23 October 2016].
[ONLINE] Available at: http://pulsiphergames.com/gamedesign/GameCreationProcess.htm. [Accessed 16 Oct 2015].

All images provided were created during the development of Snack Attack.