**Tech Design Document for:**

**Burrito Bandit**

Version # 1.2

© 2017

Table of Contents

[Overview 4](#_Toc472885114)

[Coding Standards 4](#_Toc472885115)

[Naming Conventions 4](#_Toc472885116)

[Class Names 4](#_Toc472885117)

[Method and Function Names 4](#_Toc472885118)

[Method Argument Names 4](#_Toc472885119)

[Global Variables 4](#_Toc472885120)

[Static Variables 4](#_Toc472885121)

[Constant Variables 4](#_Toc472885122)

[Enum Names 4](#_Toc472885123)

[Comments 4](#_Toc472885124)

[Code Length of Methods 4](#_Toc472885125)

[Code Length of Lines 4](#_Toc472885126)

[Brace formatting and If Then Else Formatting 5](#_Toc472885127)

[Integration plan 5](#_Toc472885128)

[Development Plan 5](#_Toc472885129)

[Milestone Deliverables 5](#_Toc472885130)

[Pre-Alpha 5](#_Toc472885131)

[Alpha 5](#_Toc472885132)

[Beta 5](#_Toc472885133)

[Release 6](#_Toc472885134)

[System Architecture 6](#_Toc472885135)

[Module Break Down 6](#_Toc472885136)

[Game Controller 6](#_Toc472885137)

[Level Controller 6](#_Toc472885138)

[Player Controller 6](#_Toc472885139)

[AI Controller 6](#_Toc472885140)

[Input 6](#_Toc472885141)

[Projectiles 7](#_Toc472885142)

[Power Ups 7](#_Toc472885143)

[Event Triggers 7](#_Toc472885144)

[Assets Folder Hierarchy 7](#_Toc472885145)

[Testing Plan 7](#_Toc472885146)

[Testing 7](#_Toc472885147)

[Internal 7](#_Toc472885148)

[External 7](#_Toc472885149)

[Proof of concept 7](#_Toc472885150)

[Pre-Alpha 7](#_Toc472885151)

[Alpha 8](#_Toc472885152)

[Beta 8](#_Toc472885153)

[Bugs 8](#_Toc472885154)

[Tracking 8](#_Toc472885155)

[Reporting 8](#_Toc472885156)

[Handling 8](#_Toc472885157)

[Player Activity Map 8](#_Toc472885158)

# Overview

This doc will serve as a Guideline for coding standards and desired results/requirements for the project.

# Coding Standards

## Naming Conventions

### Class Names

Will be in Frist letter of a word in Upper case, otherwise lower case, with no spacing.

### Method and Function Names

Will have the first letter in lower case but, further words start with uppercase. No spacing. Try to make the method behavior clear in the name.

### Method Argument Names

Will be in lower case with “\_” spacing.

### Global Variables

Global vars will have the prefix g. i.e. g\_name

### Static Variables

Static vars will have the prefix s. i.e.s\_name

### Constant Variables

Will be in all caps with “\_” spacing

### Enum Names

Will be in all caps with “\_” spacing

## Comments

Don’t forget to provide comments, typically enough to describe the relevant behavior.

## Code Length of Methods

Ideally keep the number of lines down to a page or two if you can help it. If delegating sections of code could have performance de-optimizations, make sure to use comments to help split up the sections of the method.

## Code Length of Lines

If a line of code threatens to go far right of the screen, format it to occupy more than one line. Remember,typically, the compiler only cares about “;” and not the return character.

## Brace formatting and If Then Else Formatting

If() //comment about if statement

{

}

Else //comment about if statement alternate

{

}// shorthand comment about if statement, denoting the end.

# Integration plan

We’ll be using Git to manage source control.

# Development Plan

# Milestone Deliverables

## Pre-Alpha

* Game must be able load a working main menu template.
* Main game includes a controllable player.
* Scene transition can be done.
* Documents are mostly completed.

## Alpha

* Game must have working UI’s, with functioning settings.
* Saving to information file implemented.
* Sounds added.
* Main game includes a controllable player that can collect power ups, and kill enemies with projectiles.
* Enemies must be spawn able.
* Score must be increasable.
* Power ups give benefits.
* Event triggers’ at least a boss battle and level transition.
* Player can win and lose, with appropriate cut scene.
* Story has been polished.

## Beta

* Levels have been fleshed out.
* Various features have all been implemented.
* Dialog has been all written and implemented.
* Feedback forms prepared.

## Release

* Feedback has been considered/Implemented.
* Bugs are at least unnoticeable, if not no existent.
* Polish.

# System Architecture

The majority of the architecture will be an existing engine, called Unity.

# Module Break Down

## Game Controller

* Manage game time (pause)
* Record score
* Handle scene transition
* Manage UI
* Handle non-player entity specific inputs
* Handle player health

## Level Controller

* Handle spawning
* Handle triggers

## Player Controller

* Handle player movement
* Handle player attacks
* Handle player animations

## AI Controller

* Handle NPC taking damage
* Handle NPC movement
* Handle NPC animations
* Handle NPC attacks

## Input

* Navigate thru UI
* Pause game
* Move player character
* Aim player character
* Fire projectiles

## Projectiles

* Note who/what they can hit
* Can call damage events on hit
* Self-destruct if enough time passes

## Power Ups

* Simply note their type
* Possibly self-destruct over time if appropriate

## Event Triggers

* Includes: dying, hurt, and enter area
* Can cause spawning, AI behavior changes, and scene transition

# Assets Folder Hierarchy

# Testing Plan

## Testing

### Internal

### External

### Proof of concept

### Pre-Alpha

### Alpha

### Beta

## Bugs

### Tracking

### Reporting

### Handling

# Player Activity Map

Loading

Main Menu

Settings

Intro

Main Game

In-Game Menu

Application

Load Level

Main Game

Explore

Fight Enemies

Pickup power ups

Trigger Event

In-Game Menu