# Changelog

|  |  |  |
| --- | --- | --- |
| **Date** |  | **Changes** |
| 16-04-18 |  | First copy of the technical design |
| 17-04-18 |  | Added overview & dev point of view |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Overview

Bike dash - You’ll create your own bike in the customization screen, with it’s own engine, color etc. And then you’ll race against each other and use items to stop each other. It’s tournament style, so the 1st and 2nd place will move forward in the competition and in the last round the winner wins the bike that they created.

# Development point of view of the project:

It’s a 3D racer game with customizable bikes and usable items to slow down your oppononents.

# Requirements

## Software

|  |  |  |
| --- | --- | --- |
| **What?** |  | **Why?** |
| Unity |  | Engine of choice |
| Photoshop |  | Easy to import and update |
| Maya 2018/ 3DS Max 2018 |  | Creates .fbx, which is easy to import |

## Hardware

|  |  |  |
| --- | --- | --- |
| **What?** |  | **Why?** |
| Laptop/Computer |  | We need a place to work on |
| Power outlet |  | We need power for our laptops/computers |
| Internet connection |  | We need to look up things |

# Target

## Platform

We are making this project to be able to run on a desktop ( Windows / Mac / Linux )

## Minimal device

OS: Windows Vista SP1+, Mac OS X 10.9+, Ubuntu 12.04+, SteamOS+.

Graphics card with DX10 (shader model 4.0) capabilities.

CPU: SSE2 instruction set support.

## Recommended device

A modern desktop/laptop with internet connectivity.

## Engine

We’ll use Unity because it’s easy to use and we’re coding in C#.

# Project Rules

## Folder structure

|  |  |  |
| --- | --- | --- |
| **Folder** |  | **What goes in** |
| Scripts |  | All scripts that are being made, and used go in here. Also in a sub folder with the usage |
| Assets |  | All art that comes in go in here, in a sub folder for location and usage |
| Textures |  | All textures go in here, in a sub folder for the object |
| Prefabs |  | All prefabs go in here |
| Animations |  | All animations go here, with a sub folder of usage |
| Music |  | All music go in here, with a subfolder for sounds ( like jumping ) and a folder for the type of music |
| Levels |  | Every level is in their own sub folder |

## **File types**

|  |  |  |
| --- | --- | --- |
| **File types** |  | **Purpose** |
| .wav |  | Sound/music |
| .fbx |  | So textures are already on it |
| .cs |  | All classes |
| .psd/png |  | .psd is for textures, .png is for alphas |

## Naming

**Word Choice**

Do choose easily readable identifier names.

For example, a property named ***HorizontalAlignment*** is more English-readable than ***AlignmentHorizontal***.

Favor readability over brevity.

The property name ***CanScrollHorizontally*** is better than ***ScrollableX*** (an obscure reference to the X-axis).

Using Abbreviations and Acronyms

**Do not** use abbreviations or contractions as part of identifier names.

For example, use ***GetWindow*** rather than ***GetWin***.

**Do not** use any acronyms that are not widely accepted, and even if they are, only when necessary.

Avoiding Language-Specific Names

**Do** use semantically interesting names rather than language-specific keywords for type names.

For example, ***GetLength*** is a better name than ***GetInt***.

**Do** use a generic CLR type name, rather than a language-specific name, in the rare cases when an identifier has no semantic meaning beyond its type

use ***PascalCasing*** for all public member, type, and namespace names consisting of multiple words.

use ***camelCasing*** for parameter names.

**Do not** assume that all programming languages are case sensitive. They are not. Names cannot differ by case alone.

## Version control

We use git and have a repository at [www.github.com/repo-name](http://www.github.com/repo-name)

## How to contribute

Download the project.

We have one master branch with all the work in it.

You clone the project to your own laptop/computer.

If you want to work on the project you need to branch off of the master branch with the name of the thing you’re working on.

If you want to commit to the repository you first merge with the master branch so you have everything, including your own.

Make sure there are no bugs or things that conflict with the master before you commit.

Write down clearly what the update is, and what it does.

When you’re completely done you’ll make a pull request.

The administrator will take a look, and if everything is good it will be merged with the master branch.

If not, you’ll need to update some stuff. ( The admin will tell you what’s wrong )

# Technical choices

## Rendering and view

Forward rendering path, because it’s the cheapest for mobile.

Gamma color space because it makes the mobile game look better.

The game will have a 2D side scrolling perspective view.

## Physics

it’s a 2D game, with no ragdolls, no physics because everything is done through code/animations.

## Scene management

We’ll have 2 scenes, the menu and the game. The load screen will be stacked on the game scene so it will load in the background.

## AI

We’ll use a behavior tree to check every possibility that the AI can make, and choose an appropriate option.

## Manager objects

We use managers, like an Audio manager that takes care of the audio.

|  |  |  |
| --- | --- | --- |
| **Manager** |  | **Responsibility** |
| Audio |  | Plays every sound/music |
| Level |  | Keeps track of the objects in the game |
| Score |  | Keeps track of the score in the game |

Everything that’s not in this list will be a scriptable object.

## Interface

We’ll use the basic Unity UI to make our GUI.

Everything can be clicked on it and the name interprets what it does. ( play goes to play mode, options opens an options screen etc. )

## Players

The game is a mobile game so, no LAN or split screen etc.

The player can play and interact with everything on screen with their fingers.

You aren’t able to change the controls since it’s a mobile game and the only input you have are your fingers.

## Data storage

All score will be saved into a small database with the levels that you’ve unlocked and scores.

There will only be a local highscore, so no competing against other payers

## Cutscenes

There will be no cutscenes in the game.