Abstract

Ah My 2 Legs is a physics based, 2D platformer, where the players control a prisoner who lost his legs and must escape jail. They do this by launching themselves like a slingshot through levels, to collect a key and reach the end gate.

Ah my 2 Legs  
TECHNICAL DESIGN DOCUMENT

Developer: Daniel Sohler

Contents

[About 3](#_Toc51679645)

[Change Log 4](#_Toc51679646)

[Team Members 4](#_Toc51679647)

[Development Environment 5](#_Toc51679648)

[Software Requirements 5](#_Toc51679649)

[Accounts 6](#_Toc51679650)

[Third Party Libraries 7](#_Toc51679651)

[Version Control 8](#_Toc51679652)

[Repository 8](#_Toc51679653)

[Contributors 8](#_Toc51679654)

[Commit Message Format: 8](#_Toc51679655)

[Target Platform 9](#_Toc51679656)

[<Platform> 9](#_Toc51679657)

[<Platform> Limitations 9](#_Toc51679658)

[Minimum <Platform> Specs 9](#_Toc51679659)

[Release Build Instructions 9](#_Toc51679660)

[Deliverables 9](#_Toc51679661)

[Controls 10](#_Toc51679662)

[Keyboard / Mouse 10](#_Toc51679663)

[Controller 10](#_Toc51679664)

[Mobile / Touch 10](#_Toc51679665)

[Custom Game Systems 10](#_Toc51679666)

[Coding Standards 10](#_Toc51679667)

[Coding Standards Enforcement 10](#_Toc51679668)

[Technical Goals and Challenges 11](#_Toc51679669)

[Technical Goals: 11](#_Toc51679670)

[Technical Risks: 11](#_Toc51679671)

[Risk Avoidance 11](#_Toc51679672)

# About

Ah my 2 Legs is a 2D physics-based platformer, where the player is a mutilated prisoner, using a slingshot to escape the prison. Each level they collect keys, while avoid dangerous obstacles, and use the key to open the exit door. The player must also manage an energy meter, which limits how much they can slingshot themselves.

The purpose of this document is to detail and record the development process of creation of scripts and functionality of the game. It also involves specific game information about its availability and practices ect. Version control, Coding Standards, technical guidelines.

# Change Log

Updates made to the document should be described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date of change | Description |
| 0.0.0 | AIE | DD/MM/202X | Initial Template created |
| 0.0.1 | Daniel Sohler | 16/06/2021 | Added information too ‘Abstract’, ‘About’ |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Development Environment

This section outlines the required software and systems required for development of this project.

## Software Requirements

The below table outlines the software requirements for development of this project. Developers contributing to the project are required to use the approved software outlined below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Version | License | Used By | Used For |
| Unity 3D | 2019.3.6f1 | Education | Programmers, Designers, Artists (On Campus) | Development of Game |
| Unity 3D | 2019.3.6f1 | Free | Programmers, Designers, Artists  (At Home) | Development of Game |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Libraries

Unity/Unreal comes with a default collection of plugins, tools and assets. Its plausible, and often encouraged to pull in additional assets, tools, plugins or scripts etc. developed by a 3rd party. Identify both engine and system libraries used in the project, and especially any 3rd party ones used, including licensing information on its usage.

|  |  |  |
| --- | --- | --- |
| Asset/Library/Package name | License | Used For |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Version Control

### Repository

<insert link to GIT repo>

### Contributors

* <List team members (git usernames)>

## Commit Message Format

Standard commit message will include the following information:

Please update, add or remove any details that you identify as important to include in commit message.

* **Type**: Represents the type of change, often the “Type” can be inferred based on the associated ticket in your project management tool, which may include: FIX, FEATURE, REFACTOR, DOC, TEST etc.
* **Scope:** Refers to the area of the project being changed, could refer to things like (menu) (inventory) (save\_system) (level) (controls) etc. Scopes may change throughout development but can be broadly identified. Outline the scopes below that seem suitable for your project
  + MENU
  + LEVEL
  + **…**
* **TaskId:** Id of the associated ticket representing the change.
* **Summary:** A short description of what has been changed.

**Format:**

|  |
| --- |
| Type (scope) : TaskId : Summary |

**Examples:**

|  |
| --- |
| Feature (menu) : #1302 : Added Exit button to main menu |
| Fix (menu) : #1395 : Updated button prefab with so that hover works on web builds |
| Feature (sandbox) : #1129 : Added rock asset to test scene, Created Rock prefab |
| <add your own examples> |
| <add your own examples> |
| <add your own examples> |

# Game Overview

## Description

This is a general description of gameplay. It could be taken straight from the GDD.

## Genre

Identify the game’s genre

## Perspective

Identify the basic perspective of play through the game (e.g top-down, First-person, Third-person, isometric, 2D side-scrolling, etc). This helps provide context for what is being displayed.

## Target Platforms

This project will be deployed to the following platforms:

* Windows / PC
* WebGL / Browser
* …

## <Platform>

Duplicate this section for each desired platform

### <Platform> Limitations

Outline <platform> limitations, provide short description of the limitation. could include:

* Available inputs (keyboard, mouse, touch, controllers etc)
* Performance constraints (max number of particles, game objects etc)

### Minimum <Platform> Specs

Outline the expected minimum system requirements required to run the project in release build.  
The minimum/maximum specs should consider target audience system specs and drive both technical and non-technical design decisions to ensure project runs on specified devices.

* System spec
* System spec

## Feature List

The project’s features include:

This is just a list – Details of feature implementation come in later sections.

* Feature 1
* Feature 2
* Feature 3
* etc

# Game Flow & Structure

This section of the document outlines the high-level structure and order of play for the project.

Game Modes & Handling  
This is a technical description of how modes are handled in the project. At what point are modes selected (if at all)? What scripts contain the game mode information, etc? Still cover this even if you have only one mode, as there will always effectively be ‘game mode’ information executed.

* Mode handling point 1
* Mode handling point 2
* Etc.

Game Mode - <Name of Mode>   
This section repeats for every mode in the game.

Description   
Basic concise description of how the game mode works.

Objectives   
These should ideally be actual objectives in the game mode, rather than general ones like “earn points”. We are defining technical criteria for objectives, not ‘good play’.

Objective Tracking   
Describe how objectives will be handled technically in this mode. If it is handled by the event handler, simply make that clear and reference that.

Mission / Level Structure  
This section identifies the structure of gameplay across levels / missions.

Overview of structure  
This is effectively a description of the structure of gameplay, and how the player goes through it. It should be a concise series of bullet points, possibly augmented with a diagram.

Gameplay Loops   
This section should showcase all gameplay loops, both for core gameplay, and any meta-loops too. It is vital the programming dept. understands what repeats, and what drives what so they can set up code structure to work with this. These can be taken from the GDD.

<Name of Loop 1>   
Add a few words to provide context for what it is, and of course the image of the loop.

<Name of Loop 2>   
Add a few words to provide context for what it is, and of course the image of the loop.

<Name of Loop X>   
Add a few words to provide context for what it is, and of course the image of the loop.

# Gameplay Systems

This section of the document provides specifications for the systems that drive the game.

## Controls / Input

State the proposed control scheme for each platform. Include relavent diagrams and references materials to outline the control mechanisms required. It is optional to show controller mapping, it is more important you outline the connection between inputs and systems in the game.

If there are multiple areas that have a change in control scheme, outline the intended usage for each area. For example: Menu Navigation, vs Player controls. Remove / add subsections as applicable.

### Keyboard / Mouse

Describe

### Controller

Describe

### Mobile / Touch

Describe

## Game Mechanics

All gameplay mechanics are to be covered here. You may format information as you like, but ensure you cover the info for the bullet points listed under the examples below. Keep information concise – This is reference material, not justification, etc.

### <Name of Mechanic 1>

* Description of mechanic, and its purpose.
* Details of workings / rules of mechanic. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

### /// Repeat as necessary.

### <Name of Mechanic X>

* Description of mechanic, and its purpose.
* Details of workings / rules of mechanic. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

/// End repeated section.

## Custom Game Systems

This covers systems that help drive and support gameplay that may not be a mechanic itself. For example, an inventory system may co-ordinate with mechanics such as picking up and dropping items (actions). A chat system may really be just interface systems required for the actual chat mechanics to be usable by the player. You may also have some systems that do things to your game state or level that are not a mechanic, or are simply purely technical in nature.

You MUST cover event handlers and UI systems here!

### <Name of System 1>

* Description of system, and its purpose.
* Details of workings / algorithms of system. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

### /// Repeat as necessary. <Name of System X>

* Description of system, and its purpose.
* Details of workings / algorithms of system. Feel free to use diagrams, this drastically reduces the need for your text to communicate perfectly on its own.
* Inputs (things required / involved)
* Outputs (things affected / produced / outcomes)

/// End repeated section.

## 

## Physics <if applicable>

Very simply, identify how physics work in the game. If gameplay involves conscious collision of physical objects in a level, this is worth covering to ensure you understand how physics work in your game, even if that’s just out-of-the-box stuff from your game engine.

Keep this section concise. Use bullet points where possible, just ensure context is clear – Someone else should find this informative enough to know things.

## Behaviours / AI <if applicable>

Very simply, identify how any behaviours / AI work in the game. If you have instruction sets for enemies/NPCs, then this is worth covering.

Definitely include images of any behaviour trees / state machines for AI.

Also definitely create subheadings to describe behaviour states.

# Game Content

This section of the document covers content types for the project, and provides technical specifications on their usage.

## Game Environment

Provide a summary of the nature of the gameplay environments. Does everything take place in a fixed environment? Does the player move through multiple 3D levels? Provide information in a paragraph or two, and add supporting bullet-points if useful/necessary.

The rest of this section is about ensuring you are clear on the technical specs and usage of your content types.

/// Repeat as necessary – Add any supporting sub-headings or infor required, such as diagrams, etc

## <Game Content Type X>

* Concise description of content type (just to clarify) – 1 line at most.
* Location in project (actual file location / folder structure, or file(s) the content is stored in)
* Application details (tech specs / format requirements / required processes)

### List of <content type>

* <Item 1>
* <Item 2>
* etc

/// End repeated section.

# Coding Standards

Outline the coding conventions followed during the development of the project.  
You may link to existing coding standard documentation

## Coding Standards - Details

Actually bullet-list some of your programming standards. Showing images and providing examples to help clarify is good.

* What are your criteria for when to comment?
* How do you nest brackets for sections of code?

## Naming Conventions

You should have rules on how things are named to ensure consistency and ease of understanding of intent with naming.

* How do you name variables and functions?
* List any prefixes or suffixes you use for naming.

# Technical Goals & Risks

Bullet-point any Technical Goals that can be identified for the development of this project. A goal statement should identify how it is measured for success.

Example (Goals):

* Maintaining 60 FPS on min spec webgl build with desired graphics / shaders features
* Creating configurable inventory and collection system
* Saving the game at checkpoints

## Technical Goals:

* Bullet point technical goals

## Technical Risks:

* Bullet point technical risks (skill gaps, or potential over scoped features, areas of project needing additional research)

## Risk Avoidance:

For each goal / risk outline potential approaches that can be taken to minimize the risk. Area there features that could be cut / redesigned?

# Appendix A – Technical choice justifications

This appendix serves as a place to justify some of the development environment choices you made. THIS IS IMPORTANT FOR ASSESSMENT. Following the red text under each heading, you can see what information you need to show here – This is information you will find by performing some basic research on programming languages, and game engines.

## Choice of development engine:

* Justify the choice of the engine you are using for your project. This must provide logic / reasoning as to why you would use this engine.
* Have more than one reason. Reasons can be very basic and practical based on your circumstance, but at least one must connect with qualities of the development engine – Something to show you could compare it to others.

## Choice of scripting language:

* Justify the choice of the scripting language you are using for your project (c#). This must provide logic / reasoning as to why you would use this language.
* Have more than one reason. Reasons can be very basic and practical based on your circumstance (like what engine you are working with), but at least one must connect with qualities and capabilities of the scripting language – Something to show you could compare it to others.

## Choice of third-party libraries and content: (if required)

* Justify the choice of any 3rd party libraries and content used in the project. This must provide logic / reasoning as to why these were good choices to make for your project.