GAME NAME  
Production Planning   
TECHNICAL DESIGN DOCUMENT

Team Name: <Nanana>

ANY TEXT IN RED SHOULD BE REPLACED OR REMOVED – Remove this box when submitting

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

Describe the project / Game (1 paragraph)

Describe the purpose of this document (1 paragraph)

# Change Log

Updates made to the document should be described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date of change | Description |
| 0.0.0 | AIE | 22/09/2022 | Initial Template created |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Team Members

|  |  |
| --- | --- |
| Name | Role |
| John Doe | Programmer |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 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.

Any software that contributes to the direct development including planning and communication tools should be outlined below. A developer contributing to the project should have the below software available to them for use.

|  |
| --- |
| This includes tools like Microsoft Teams, Trello, HackNPlan, Git, Image / Audio editing tools, modeling tools etc. A discussion should be had with the team to ensure all areas are identified appropriately. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Version | License | Used By | Used For |
| Unity 3D | 2020.3.5f1 | Education | Programmers, Designers, Artists (On Campus) | Development of Game |
| Unity 3D | 2020.3.5f1 | Free | Programmers, Designers, Artists  (At Home) | Development of Game |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Accounts

The below table outlines any accounts that may be needed for the development of the project. An account is usually identified by 2 areas:

* **Individual**: Each developer of the project may need an individual user account, for various software or services. This includes software like Trello, HackNPlan or Git.
* **Organization**: A project or organization account is often developed for the software to integrate with other services, this includes things like Advertising / AdSense / git organizations / repos, Facebook developer account etc. An organization account is usually managed by 1 or more team members. Ownership of the account should be able to change between members.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Account/Service | License | Used By | Used For | Owner |
| Github | Free | Programmer, Art, Design | Contributing to projects hosted on github | NA |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Third Party 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. Any additional library or assets developed by the third party should be listed below.

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

# Version Control

## Repository

<insert link to GIT repo>

## Contributors

* List team members (Student and git usernames)

## Commit Message Format:

Within GIT, a commit message can contain any arbitrary text, no format is enforced. However, it is useful for commit messages on a project to follow a consistent format that includes pointed details about the changes that have been made in the commit.

A Good commit message will include the following information

* **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:  
  + **Feature:** This commit implements a new feature, makes progress, or improves a feature.
  + **Fix:** A bug has been identified, this commit relates to changes that resolve the issue
  + **Refactor:** The code, folder structure, or other parts of the project needs some adjustments to better support maintainability and addition of new features.
  + **Performance**: This commit has changes that improve the projects performance
  + **Docs:** This commit has made changes to documentation
* **Scope:** Refers to the area of the project being changed, could refer to things like (menu) (inventory) (save system) (level) (controls) etc… Scope’s may change throughout development, but can broadly identified. Outline the scopes below that seem suitable for your project
* **TaskId:** Id of the associated ticket representing the change  
  Your commit will be automatically linked to a github issue when the message contains the GitHub issue ID.
* **Summary:** A short description of what has been changed.

**Commit Message 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 |
| Docs (readme) : #1111 : Updated project summary and title |
| Performance (level) : #4913 : Improved level generation code |
| Performance (player) : #4912 : reduced texture resolution and model vertex count for the player to ensure performance on web builds |

# Target Platform

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:

* Graphics capabilities, shaders, poly counts
* 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

### Release Build Instructions

Detail the steps needed to build for the desired platform – Are there any manual steps required to prepare your project for release?

## Deliverables

A Build of the project should be generated every \_\_\_\_\_\_ and placed in the following location:

|  |
| --- |
| path to network drive folder / <BuildID> / <platform> / \* |

**Build ID**:  
When uploading the build, a sequential naming strategy should be used.  
Outline the naming convention that should be followed when updating the daily build.  
Some examples include:

* Semantic versioning: major.minor.patch  
  <https://www.geeksforgeeks.org/introduction-semantic-versioning/>
* Date Time versioning: YYYY\_MM\_DD

# Custom Game Systems

Each game is built on top of a structured system to allow for flexibility in design, and ease of changes to code. Game systems are custom to your game, so they should be described and planned for here.

Provide UML inheritance diagrams, flow charts or other to help communicate the design (if applicable)

Example Game Systems, and things that could be identified

* Controls Management:   
  The design document should outline the expected behavior for controls for your game. Here, you should document how this has been implemented, components/scripts/settings. If your controls change per platform, you want to minimize the volume of scripts and game objects that need to be changed, implementing an abstraction layer will assist with this, you should document how game objects should interact with your abstraction layer.
* Chat / Dialog system:  
  how events trigger a dialog.  
  Are there multiple screens / text to navigate through.  
  Are there options to select… how might you configure a new dialog
* Event Handling:  
  If there are events triggered within the game, are there design patterns that should be followed, eg: pressure plate opening a door?
* Attacking / Dealing damage  
  How would objects be identified as “Attackable” or as an “Attacker”
* Pickups / Collectables  
  How would a new collectable be added to the project? What tags, scripts or settings need to be updated. Do you need UI icons? 3d assets? Tooltip messages? Descriptions? Can the item be “used” how would you specify the action that is called. Document components, scripts, settings, prefabs, and steps to be taken to for implementing a new pickup/collectable.

# Coding Standards

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

## Coding Standards Enforcement

How should coding standards be enforced, code reviews? Linting tools? Any special setup or processes to follow to aid in this process?

# Technical Goals and Challenges

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?