Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Simple Ball Game  
TECHNICAL DESIGN DOCUMENT

Developer: Matthew Wakely

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. Focus on it as a product, rather than a gameplay pitch. (1 paragraph)

Describe the purpose of this document (1 paragraph)

An arcade game where the player is trying to get as high a score as possible

# Change Log

Updates made to the document should be described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date of change | Description |
| 0.0.0 | MW | 30/08/2021 | Initial Template created |
| 0.1.0 | MW | 31/8/2021 | Completed Version Control, Game Overview, Game Flow & Structure, Gameplay Systems, Game Content, Coding Standards, Technical Goals & Risks |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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 | 2020.3.14f1 | Free | Programmers, Designers, Artists  (At Home) | Development of Game |
| Visual Studio Community 2019 | 16.11.2 | Free | Programmers, Designers | Scripting |
| GitHub Desktop | 2.9.2 | Free | Programmers, Designers, Artists | Version Control |
| Microsoft Word | 2020 | Student | Programmers, Designers, Artists | Documentation |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 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 |
| Standard Asset Pack | Free | Third Person Controller |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 

## Version Control

### Repository

### <https://github.com/mdwakely/Design-Prototyping-Fundamentals-Matt>

### Contributors

* Matthew Wakely (mdwakely)
* Dan Beaumont (???)

## Commit Message Format

Standard 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: FIX, FEATURE, 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
  + GUI
* **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 to work on hover |
| Feature (level) : #1234 : Added box collider to goal prefab |

# Game Overview

## Description

## This is an arcade game where the aim is to score goals in the quickest time possible with more points given the quicker the player scores a goal. The goal posts will reset their location after each goal.

## Genre

## Arcade/Sport/Casual

## Perspective

Third-person

## Target Platforms

This project will be deployed to the following platforms:

* Windows/PC

## Windows/PC

Duplicate this section for each desired platform

### PC Limitations

* Inputs required for both mouse & keyboard, and controller
* Must ensure that the game can perform on a wide range of machines

### Minimum PC Specs

* Windows 10
* CPU: 64-bit
* GPU: Integrated graphics or dedicated graphics card
* RAM: 4GB
* Storage: 200MB

## Feature List

The project’s features include:

* Main Menu with start and quit options
* High score system
* GUI with timer, current score, and high score
* Level with OnTriggerEnter reset
* Scoring system

# Game Flow & Structure

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

Game Modes & Handling

* Main Menu
* Game Level

Game Mode – Main Menu   
Description   
A simple menu with two buttons, ‘Start’ and ‘Exit’, for getting in and out of the game.

Objectives

* Get the player in to the Level scene via a button
* Enable the player to exit the game via a button
* Track high score

Objective Tracking

* Game manager keep track of high score between game states, and move game to playing state

Game Mode – Game Level   
Description   
The player will spawn in the centre of the level with a ball in front of them, the GUI will have a timer in the top-right and a score in the top-left.

The player’s aim is to kick the ball in to a goal area to achieve a score.

The level will then reset with the goal area in a different location.

Objectives

* The player character must be able to move around the game area
* The player character must be able to move the ball around the game area
* The goal requires a box collider to trigger the score, timer reset, and level reset
* The goal is required to spawn at different locations throughout the game area
* Have a pause menu with an exit button and a resume button

Objective Tracking   
A Game Manager game object will have a script that keeps track of the Player Score and Time, spawns the goals on level reset, and move the game to the menu state.

Goal to have a script that keeps track of when a goal is scored.

Mission / Level Structure  
**Overview of structure**

* Player begins the level in the middle of a football field with a ball in front of them
* At the start of the level score is 0 and the timer begins to tick down
* The quicker the player scores a goal the higher their score is
* After scoring a goal the level will reset with the goals in a different location for the player to score again

Gameplay Loops   
**Kicking a Goal**



# Gameplay Systems

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

## Controls / Input

### **Keyboard / Mouse**

W/A/S/D – Character movement

Esc – Pause Menu

Left Mouse Button – Menu interactions

### **Controller**

Left Joystick – Character movement

Menu Button – Pause Menu

A Button – Menu Interactions

## Game Mechanics

**Player Movement**

* Enables the player to move around the level at either a walk or run, and enable them to jump
* Keyboard or controller inputs to move the player forward/backward and left/right.

**Kicking the Ball**

* A passive mechanic, if the player walks into the ball the ball will move, allowing the player to maneuver the ball towards the goals
* Rigidbody on the Ball and contact with the mesh collider of the Player Character causes the movement

## Custom Game Systems

**GUI**

* Keeps track of the timer, current player score, and the high score
* Timer counts down with seconds and milliseconds shown
* High score is in whole numbers
* A game manager will handle the script for the GUI

**Goal Reset**

* After every successful goal or timer runout, the level resets and the player and ball return to the center of the world while the goal moves location
* The goals will always face inwards towards the center of the world
* A script on the game manager will control this

## Physics

* Collision between the player and the ball
* Collision between the player and the ball and the ground
* A normal gravity to ensure that the ball can be controlled

# Game Content

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

## Game Environment

**Level**

* Single level where the game takes place
* Location: Design-Prototyping-Fundamentals-Matt > Design Fundamentals Project
* Details: Unity scene, assets, and build. A single location to simulate a football pitch

**Contents**

* Scenes: Main Menu, Level
* Scripts: for GUI, player movement, goal event, goal placement
* Materials
* Game Objects to create level
* Player Character

# Coding Standards

## Coding Standards - Details

* Comment descriptions of what code is doing
* Commenting to be used to describe why code is disused
* Disused code to be commented out until code is finalised
* Brackets to be nested on the same line as they open on

e.g. void BracketHere()

{

}

* Unique functions to be placed before start or update functions
* Place same Data types together at the start of the code, e.g. all floats together, all int together, etc.

## Naming Conventions

* Use Camelcase for variables; likeThis
* Use capitals for classes: LikeThis
* Names should be clear as to their function: ExplainsFunction()

# Technical Goals & Risks

## Technical Goals:

* Have a countdown timer that resets
* Have a score that goes up and keeps the score throughout the level reset
* Have a High Score system that saves across game sessions
* Have a randomly spawning goal area

## Technical Risks:

* Saving High Score data may be beyond skill
* Goal spawning needs to work correctly so that the game can function properly

## Risk Avoidance:

* If the level reset function does not work correctly then a simple return to menu and restarting the level can be a fix, not ideal solution
* If goal spawning does not work as intended, a level system with the goals placed in different locations could solve the problem

# Appendix A – Technical choice justifications

## Choice of development engine:

**Unity**

* A good engine to create games in 3D with integration to script in Visual Studio
* Uses C# scripting
* Good to learn in due to learning C# scripting as well as general game logic, although not directly this can translate to other engines such as Unreal.

## Choice of scripting language:

**C#**

* This is the code language that Unity works in, it will allow for specific functions to be created to enable the game to work, e.g. a function to start the game level
* Unlike a visual coding method such as the Unreal engine uses, C# can be more ordered. For example a visual coding method may end up having a very large and complex system to look at, whereas a C# script will all be located in the single file and has intellisense to assist with navigating the script.

## Choice of third-party libraries and content:

**Unity Asset Pack**

* This package has a simple Third-Person Character controller that will work for this project allowing the focus to be on scripting that makes the game function and the GUI function.