# Character Controls and HUD Systems TDD

## V\_00.01

## Author: Finn Varvell

# Introduction

## Rationale

This TDD is for adding elements of a HUD into the game, including status bars, a minimap and compass and a damage flash and death screen. Other elements of controlling the player such as crouching and sprinting, keybinds and taking damage/dying will also be functional.

## Background

The HUD in a game is the display on-screen as the player plays the game that includes all the necessary information at the time for example health, score, available skills and status effects. The exact elements of the HUD differ depending on the type of game, but almost every game requires some kind of displayed information that the player can see to follow what’s happening in the game.

## Terminology

HUD – Heads Up Display

TDD – Technical Design Document

GUI – Graphical User Interface

UML – Unified Modeling Language

HP – Health points/Hit points

MP – Mana points

## Non-Goals

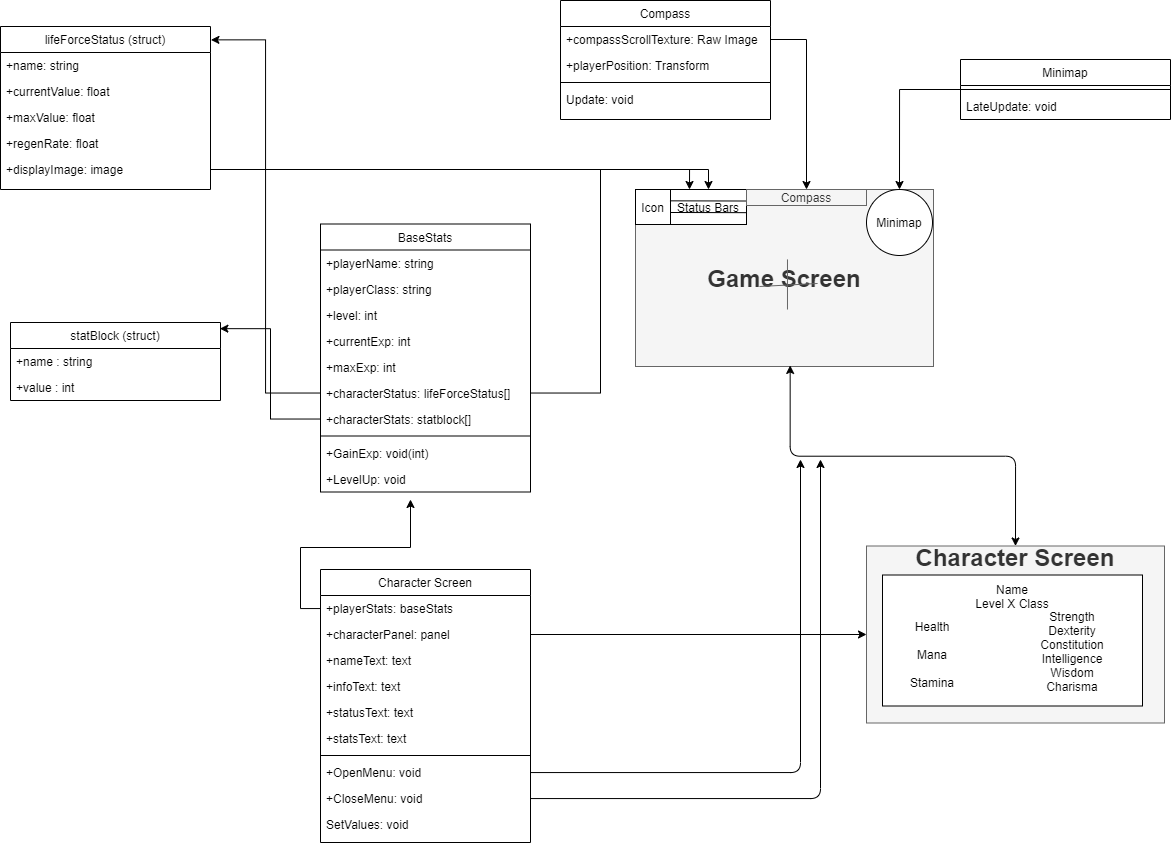
* Player able to show and hide different HUD elements from options

## Proposed Design

* Player Icon
  + Displays a front view of the character next to the status bars
* Status bars
  + Bars for health, mana and stamina coloured differently to show which is which
  + On values changing (e.g. taking damage) amount filled in bars changes to reflect new value
  + Bars refill at different rates
* Minimap
  + Minimap displayed showing player from high above
  + Location markers on minimap for shops and important areas
  + Markers for player and enemies on minimap
  + Compass displayed next to minimap showing which direction the player is facing
* Gameplay
  + Player can move and look around and has controlled based off keybinds
  + Player can crouch and sprint to change speeds
  + Sprinting uses up stamina and player cannot sprint if at zero stamina
  + Screen flashes red when taking damage
  + Death screen
    - When player health hits 0, player dies and cannot move anymore
    - Upon dying a screen comes up that allows the player to respawn or quit
    - Respawning sets the character back to a location and sets status values back to full
  + Crosshair in center of screen

# System Architecture

If the design consists of a collaboration between multiple large-scale components, list those components here — or better, include a diagram [UML].



## Data types

### BaseStats

* String for player name and class
  + Strings are arrays of characters that make up text.
  + The name of the player and their class will be stored in strings to be shown on the character screen when it is opened
* Ints for level, current and max exp
  + Integers are numerical values that are whole numbers rather than decimals. Because level and experience values will never need decimal points, intscan be used instead of floats
  + The gainExp function will increase exp, and if it reaches the max exp value, it will call levelUp, resetting exp and increasing level
* Structs for LifeForceStatus and Statblock
  + A struct is a set of variables grouped together. When an array of them is used, each element of the array will contain every variable in them
  + An array of LifeForceStatus structs will be used for health, mana, and stamina, with each element holding a name as a string, a current,max and regen values as floats and an image for the bar on the hud.
  + An array of Statblock structs will be used for each of the six stats. Each stat only needs a string for name name and an int for value.
  + By making the structs serializable, their values can be edited in the unity inspector
* Images for each value (Health, mana, stamina)
  + In each LifeForceStatus struct, an image will be included. This is a reference to the bars on the HUD, and can be used to change how they look.
  + The fill amount of the referenced image will be changed based on the current value of that element, making the bars go up and down to show how much the player has.

## Data Model

* Save and load – saving the game stores values from baseStats such as name, class and each status and stat block to be loaded when the game is reopened

## Interface/API Definitions

Describe how the various components talk to each other. URL and the format of the data and parameters used.

Libraries used:

UnityEngine.SceneManagement – Allows SceneManager to be used, which can call functions such as LoadScene, which changes the scene to another one based on its parameter.

UnityEngine.UI – Allows references to the Unity UI such as the text, buttons and dropdowns/sliders/toggles. All the functions of them can then be used, and the values can be changed.

## Impact

* Performance – doing things to reduce the performance cost of running the game
  + Running events that only need to be run once a single time rather than every frame,
  + Upon opening character screen, all the text elements are set just once when it is opened rather than every frame, as they will not change while the game is paused.
  + Coroutines to wait for responses over multiple frames
* Security
  + Public variables vs private variables – Public can be accessed outside of the class and changed from the editor and private can only be accessed from within the class. Also, protected variables are the same as private but can be accessed by classes inheriting from them.
  + Saving – When saving character data like level and stats binary should be used because it can save to any location with any name, making it harder to find, and it is also more difficult to edit.

## Risks

If there are any risks or unknowns, list them here. Also, if there is additional research to be done, mention that as well.

* There are no ways to change the players stats now, so the character screen will always show the same thing, meaning there is no need for it to get the information from the player in the first place

## Alternatives

If there are other potential solutions which were considered and rejected, list them here, as well as the reason why they were not chosen.

* Allowing the player to change their stats from a menu would show that the character screen did update based on the characters stats, but there isn’t much point adding that because when the customisation part is done the player will be able to choose from there anyway.

# System Testing

## Testing

Show progress, Error reports and explain fixes you used.

**Minimap HUD** – I wanted to make the minimap always face north but because the camera was parented to the player it also rotated as the player turned around. To stop it from doing that, I added a script to the minimap to make it reset its rotation every frame, but then there was a problem where the minimap kept shaking as it rotated and set back. After trying a few things putting the code to reset rotation in late update instead of update fixed the problem.

**Location markers** – Adding the location markers to everything I wanted a symbol for on the minimap was a bit slow, especially after I added NPCs later, so I tried to make a script that would create them for me given a sprite, but because of the different sizes of sprites, some ended up being huge and covering the whole minimap and some were too small to see. In the end I just added some code to some scripts such as the NPC shop that would add a specific symbol to the minimap at a certain scale when the game started.