Module: ENG1/ASSESSMENT2

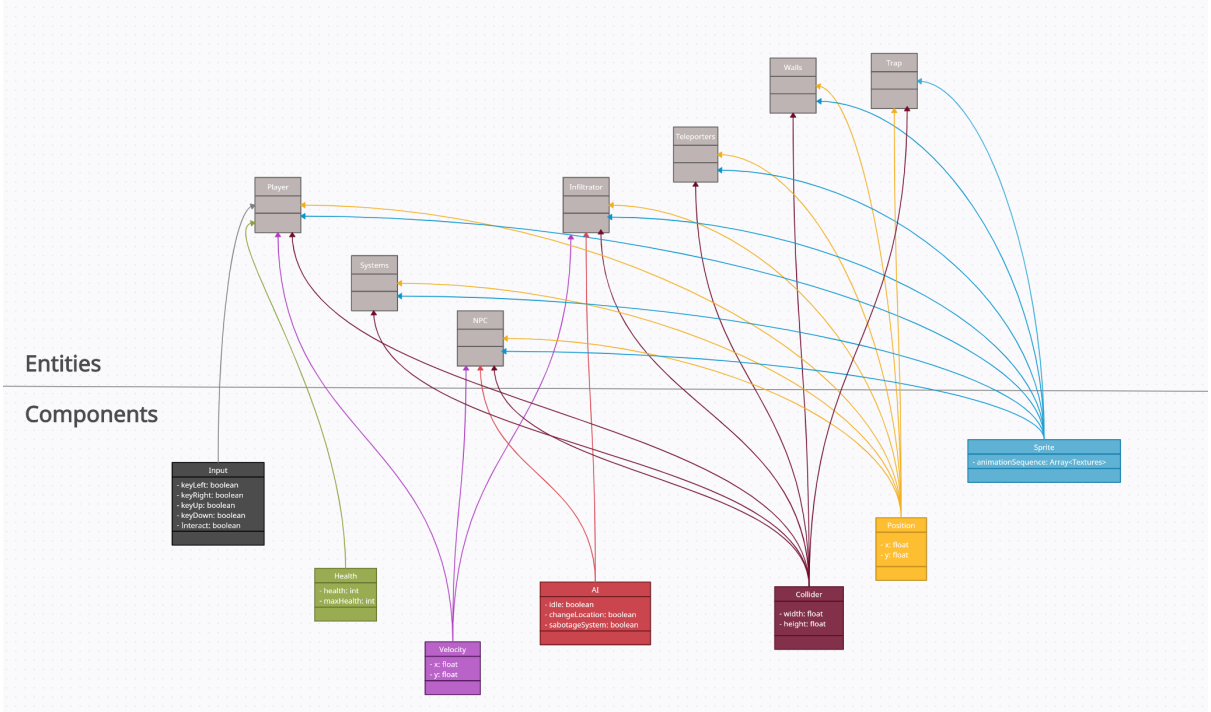
Title: Architecture

1. **Charles Stubbs**
2. **Annabelle Partis**
3. **Kieran Ashton**
4. **Yu Li**
5. **George Tassou**
6. **Alex Shore**

Our choice to present the abstract architecture of our game is an Entity Component System. This is used to define what the components within the game are made of and where they share elements with each other. The graph shows the components of Auber as well as the Entities, split across the middle, colours of lines are consistent with the colours of the components, allowing for easy reading of the document. This can be used in conjunction with the later mentioned concrete architecture to create a complete overview of how Auber will be implemented.

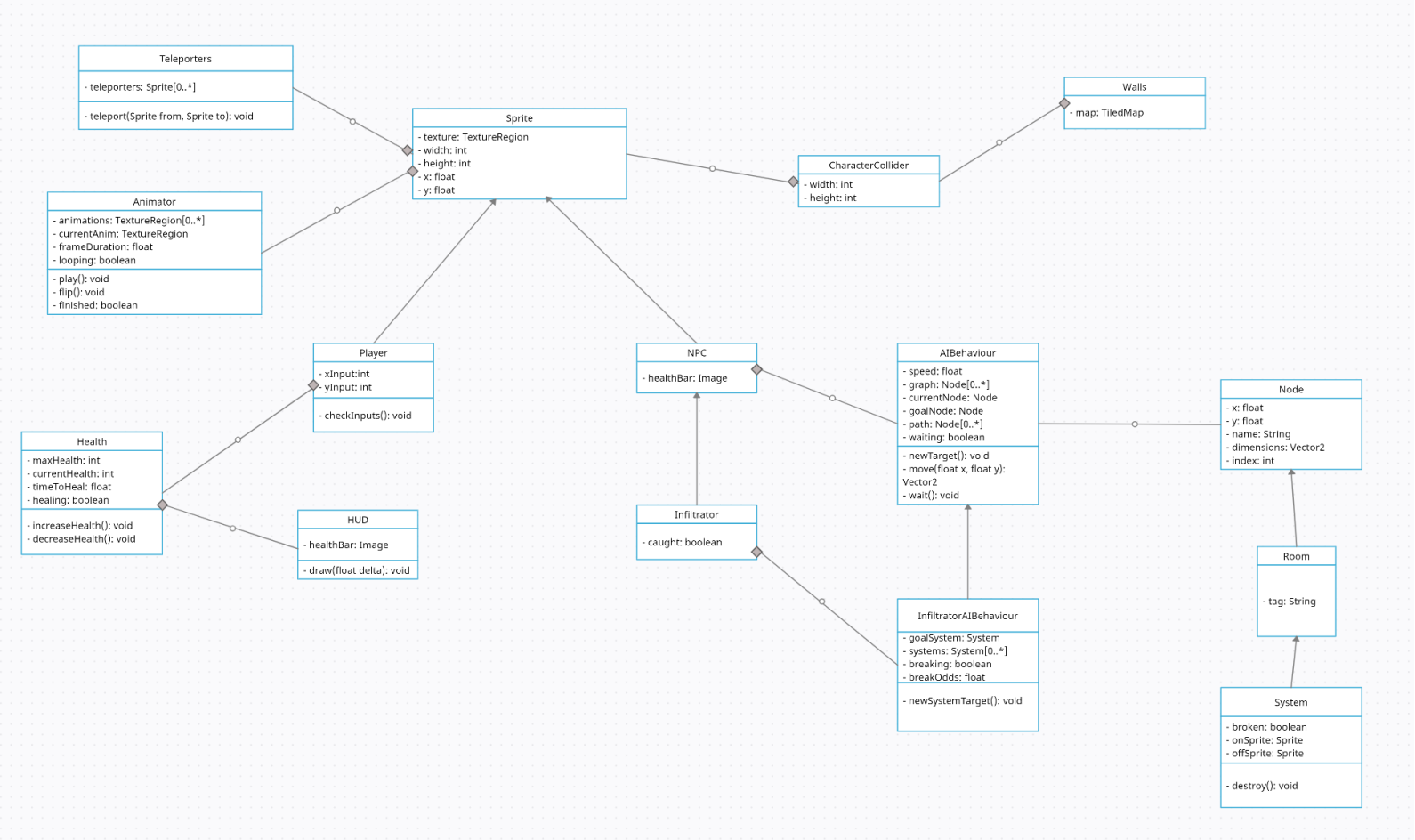
The abstract architecture below was created using Creately, a web based tool. (https://app.creately.com/)

Abstract Architecture: Entity Component System



[For Clear Diagram see website link]

Concrete Architecture: Class Diagram



[For clear diagram see website for link]

The abstract diagram focuses mainly on the entities within the game that the player will interact with, for example the NPC's and Infiltrators that make up the core gameplay and the player interacts with these entities throughout.

Within the diagram we can see there are some entities with unique or rarer components, input is naturally unique to the player and therefore is handled by the player class itself in the class diagram. Health is also unique to the player entity but in considering an infiltrator power could be to also possess health we decided to separate health from the player class, health would also need to be visible to the player on the hud and so in the class diagram it has an association to the hud class.

As the infiltrators require unique skills, the AI component has different requirements for the different types of NPCs. Differing AI configurations can be facilitated with inherited classes and then associations with the relevant sprite types.

Everything requires a position to allow the game engine to draw the relevant textures and compute the possible collisions. To facilitate this requirement, the Sprite class

contains the grid position reference which is then inherited by all the other entity classes that need one. The Sprite class is considered the core of all entities within the game's implementation.

Also inherited along with the sprite class is the association to the CharacterCollider class which is used to establish the collision boundary around an entity's position such that the full dimensions of the texture are encompassed to ensure everything collides as it should.

The one exception to this is the Walls class which also has its own association to the CharacterCollider but is instead comprised of a TiledMap, an inbuilt class of the libgdx engine, allowing us to not worry so much about the specific implementation of boundaries ensuring that the player safely remains within the play area.

The concrete architecture also diverges slightly from the abstract one with some small implementation details.

We decided to use a node system to direct the AI around the map which has its own class with two inheriting classes. The first inheriting class is the Room class. This is used to distinguish between normal nodes and nodes in rooms on which it would be more normal for the AI to exhibit its waiting behaviour rather than randomly stopping and gathering around invisible points in corridors. The second inheriting class once again inherits from the Room class to further distinguish nodes which are in front of the key systems of the ship. These are the main focus of the game, with the players main goal being to protect the systems from these infiltrators. This allows infiltrators to decide upon random targets anywhere on the map and approach them to exhibit their sabotaging behaviour, which may also be punctuated by waiting for a short period to attempt to emulate the normal NPC's.

The other divergence is the Teleporters class. Rather than being a part of the systems entity and thus being vulnerable to sabotage by infiltrators they are their own class, all of them are represented in the one class however as they are all linked to one another and so need access to the other instances. This decision was made due to the importance of the teleporters to the gameplay and player's mobility the removal of which would result in a less engaging and satisfying user experience where the player feels punished for not being able to be in multiple parts of the map at the same time.