Design Rationale for Requirement 5: Randomised Generation

Explanation of system

Randomly generates trees, enemies, and mystery blocks on world creation, rather than having them in fixed, hard-coded locations. This also includes creating the mystery blocks themselves as they were not an existing feature from Assignment 1/2 or Assignment 3 requirements. Mystery blocks can broken/opened by Mario using a Wrench, and will drop some rewards, such as items or coins.

Design choices ie. Choices made to follow SOLID principles

Single Responsibility Principle:

We will have a function for various objects of a specified type, mainly actors or grounds. These functions will be in the map class, as they will be used on only map objects, and are not needed at a higher or lower level. This adheres to the SRP as the functions which relate to maps are being added to the maps class, which maintains the single purpose of the maps class, which is to create and manage maps. Additionally, the mystery blocks will be made as a new type of ground, with information such as the items contained stored the new class. This also supports SRP as all of the information and functions needed for the mystery blocks to function will be in the one class, with no extra unrelated functionality in the class, and no mystery block specific functionality outside of it.

Open/Closed Principle:

As hinted at previously, the random generation function will be made such that they accept a parameter for the type of object (subclasses of actor or ground) to generate, so that they can be easily extended for new objects added in the future, therefore supporting the OCP. Additionally, the mystery blocks will extend the already existing ground class, and be implemented using already extensible code, therefore maintaining OCP. The blocks should also be able to give out various types of rewards, particularly items, which will allow any future items to be easily added to the 'loot pool'.

Liskov Substitution Principle:

LSP is followed as mystery blocks will be implemented by extending the already existing ground superclass, and therefore must include the necessary ground traits. Therefore, the mystery blocks must be replaceable by any other type of ground for the application to work.

Interface Segregation Principle:

This implementation should not require the addition of any new interfaces as there are not other similar classes to the mystery blocks that require similar functionality outside of the base ground functionality which is used by all ground subclasses in its entirety.

Dependency Inversion Principle:

The DIP is adhered to as the mystery blocks class will extend the ground superclass, which is also extended by other subclasses of ground that share some necessary functions. The mystery block specific functions will remain in only the mystery block class as there are no other classes that require this unique functionality.