# Title:

Randomised Generation

# Description:

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.

# Explanation as to why it adheres to the SOLID Principles:

## Single Responsibility Principle:

We will have a function for randomly generating actors, and a function for randomly generating ground of specified types. 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 functions will be made such that they accept a parameter for the type of actor or ground to generate, so that they can be used with any type of actor or ground added in the future, supporting the OCP by making the code easily extensible in the future. 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. The already established resettable interface will be used for mystery blocks, which is already implemented for numerous other classes in a way that supports ISP as it contains no functionality that is not required for any implementing class.

## 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.

|  |  |
| --- | --- |
| **Requirements** | **Features (HOW) / Your Approach/ Answer** |
| **Must use at least 2 classes from the engine package**  Mystery blocks will use Ground and Item, Random generation will use actor and ground | Mystery blocks will extend the ground class from the engine as they will be a new type of ground. The mystery blocks class will contain an inventory of various Item subclasses that it may drop upon being destroyed.  The random generation functions will take parameters of actor or ground subclasses, in order to determine what object is to be randomly spawned. |
| **Must use/reuse at least one existing feature**  Random generation will use Trees/Sprouts, and enemies/ Goomba and Koopa. Mystery blocks will use Items/Super Mushroom and Power Star. | The random generation functions will generate the previously implemented Tree and/or Sprout objects, as well as the existing enemy subclasses Goomba and/or Koopa.  The mystery blocks will be able to drop existing items such as Super Mushrooms, and Power Stars, as well as coins when destroyed. |
| **Must use existing or create new abstractions**  Mystery blocks will use the existing abstractions of Ground and Action, and implement the interface Resettable | Mystery blocks will be a new type of ground, and hence will extend the ground class. Part of its implementation will be allowing Mario to break it, which will require a new BreakBlockAction extending the Action class. It will also implement the Resettable interface to have some effects when the game is reset. |
| **Must use existing or create new capabilities**  Mystery blocks will use the existing HAS\_WRENCH capability, Random generation will use the existing DIRT capability. | Mystery blocks will only be destroyable when Mario has the wrench, hence will use the existing HAS\_WRENCH capability.  Random generation will only occur on empty ground spaces, so as to not disturb the default maps and core locations, such as the safe zone floors or existing actors such as Toad. For this it will use the DIRT capability to check if the ground to spawn on is valid. |