## 3. Specific Requirements

### 3.1 UI/UX (Functional Requirement)

#### 3.1.1 General Gameplay UI

3.1.1.1 Shall provide the ability for the player to access a custom API through the in-game API Book Button. (ceveritt)

3.1.1.2 Shall provide the ability for the player to access a settings menu through the in-game Settings Button. (ceveritt)

3.1.1.3 Shall provide the capability for certain in-game objects to have read/write functionality denoted by green highlighting. (ceveritt)

3.1.1.4 Shall provide the capability for certain in-game objects to have read only functionality denoted by red highlighting. (ceveritt)

3.1.1.5 Shall provide a button that resets the level from the beginning of the level (mtuckson)

#### 3.1.2 General Dependant UI

3.1.2.1 Shall display the API Index with listing of the full PolyMorpher API. (cbatten)

3.1.2.2 Shall display the API Index with a listing of suggested API entries to be used in any specific level. (cbatten)

3.1.2.3 Shall display the entry detailing the selected function from the API Index. (cbatten)

3.1.2.4 Shall provide the ability to edit the source code of an object, selected in the General Gameplay UI stage of interaction, through the Coding Interface. (cbatten)

3.1.2.5 Shall provide the ability to insert example code from the API Book directly into the Coding Interface through a dedicated button. (cbatten)

3.1.2.6 Shall provide the ability to reset erroneous code in the Coding Interface to a stable build through a dedicated button. (cbatten)

3.1.2.7 Shall provide the ability to compile code in the Coding Interface, changing the behavior of a selected object within the game in real-time. (cbatten)

#### 3.1.3 Level Dependent UI

3.1.3.1 Shall display example codes through read only objects. (ksantos)

3.1.3.2 Shall show hints or directions on signboards. (ksantos)

3.1.3.3 Shall display the “Textbook” curriculum by breaking down the learning sections before and after each puzzle/challenge.(ksantos)

### 3.2 Assets (Functional Requirement)

#### 3.2.1 Main Character

3.2.1.1 Shall provide directional character controls through the keyboard (i.e., Keyboard/Mouse) (ddang)

3.2.1.2 Shall provide walking animations for the main character (ddang)

3.2.1.3 Shall provide attack animations for the main character (ddang)

3.2.1.4 Shall provide death animations for the main character (ddang)

3.2.1.5 Shall provide a sound effect when the main character walks (ddang)

3.2.1.6 Shall provide a sound effect when the main character attacks (ddang)

3.2.1.7 Shall provide a sound effect when the main character dies (ddang)

#### 3.2.2 Camera

3.2.2.1 Shall follow the player with a slight delay to prevent unnatural camera movements (jstokes)

#### 3.2.3 NPC

3.2.3.1 Shall provide walking animations for each NPC (mtuckson)

3.2.3.2 Shall provide attack animations for each NPC (mtuckson)

3.2.3.3 Shall provide death animations for NPCs that die (mtuckson)

3.2.3.4 Shall provide a sound effect each time the NPC attacks the player (mtuckson)

3.2.3.5 Shall provide a sound effect each time the NPC dies (mtuckson)

#### 3.2.4 General Objects

3.2.4.1 Shall provide assets in the form of obstacles such as:

- Spikes
- Crates
- Doors
- Torches (tjohnson)

3.2.5 Environment

3.2.5.1 Shall provide sprite tiles that show the player the boundaries of each level. (priley)

3.2.5.2 Shall provide repeating sprite tiles that make up the background of each level. (priley)

3.2.6 Background Music

3.2.6.1 Shall provide music for the game as the player moves through each level (tjohnson)

3.3 Lesson Sets (Functional Requirement)

3.3.1 Inheritance Puzzle

3.3.1.1 Shall provide the assets relating to a series of obstacle course challenges and vehicle parts. (cbatten, priley, ddang)

3.3.1.2 Shall provide an extensive obstacle course, requiring the player to modify a vehicle to overcome said course, while utilizing a vehicle super-class for the modifications to be inheried. (cbatten, priley, ddang)

3.3.2 Abstraction Puzzle

3.3.2.1 Shall provide the puzzle assets of plant(s) in pot/ground, plant(s) that have been cut (just the leaf), pot(s), fire, orc, plant leaf(s), lake, ice chunks, and arrows. (mtuckson, ceveritt, tjohnso)

3.3.2.2 Shall provide a puzzle where a potion must be made with 3 plants prepared different ways in order to put an enemy to sleep. (mtuckson, ceveritt, tjohnso)

3.3.2.3 Shall only allow player to advance once they have properly implemented abstraction principles in completing the levels' puzzles. (mtuckson, ceveritt, tjohnso)

3.3.3 Encapsulation Puzzle

3.3.3.1 Shall use flying eyeball creatures to instill the feeling of the player's code "being watched", helping to visualize the principle of encapsulation, keeping their code private from the peering eyes. (jstokes, ndearce, ksantos)

3.3.3.2 Shall provide a puzzle that contains:

- An enemy/gate that changes the properties of things that pass through them.
- The player would need to have a certain property to advance though the enemy/gate
- However the enemy/gate changes it because the property is public.
- The player would need to modify the object (or its parent) to hide the variable from the enemy/gate. (jstokes, ndearce, ksantos)

3.3.4 Polymorphism Puzzle

3.3.4.1 Shall use palette-swapped versions of pre-existing graphical assets to give the visual feedback of ice-cold or red-hot surfaces. (jstokes, ndearce, ksantos)

3.3.4.2 Shall apply a specific block class that can have a type of "fire" or "ice", and depending on what it's current type is, its classes would act differently. One block would be "Wood" and the other "Metal" which both would have the same code, but their type determines how the function that it uses affects it. (jstokes, ndearce, ksantos)

3.3.5 Tutorial Mechanics Puzzle

3.3.5.1 Shall provide tutorial to teach the player movement controls (mtuckson, ceveritt, tjohnso)

3.3.5.2 Shall provide tutorial to teach the player the difference between read/write objects, read objects and non editable objects. (mtuckson, ceveritt, tjohnso)

3.3.5.3 Shall provide tutorial to teach the player how to open a script (mtuckson, ceveritt, tjohnso)

3.3.5.4 Shall provide tutorial to teach the player how to use the API book (mtuckson, ceveritt, tjohnso)

3.3.5.5 Shall provide tutorial to teach the player how to compile a script (mtuckson, ceveritt, tjohnso)

3.3.5.6 Shall provide tutorial to teach the player how to handle errors in their scripts (mtuckson, ceveritt, tjohnso)

3.3.5.7 Shall provide tutorial to teach the player about reopening scripts (mtuckson, ceveritt, tjohnso)

3.3.5.8 Shall provide tutorial to teach the player how to open saved code snippets (mtuckson, ceveritt, tjohnso)

3.3.5.9 Shall provide tutorial to teach the player how to use the reset button (mtuckson, ceveritt, tjohnso)

3.3.5.10 Shall provide tutorial to teach the player how to know once they have won a level (mtuckson, ceveritt, tjohnso)

3.3.5.11 Shall provide a sample puzzle in order to do requiements 3.3.5.1 - 3.3.5.10 (mtuckson, ceveritt, tjohnso)

3.3.6 Tutorial Code Puzzle

3.3.6.1 Shall use the base assets that will be used in the rest of the game. (cbatten, priley, ddang)

3.3.6.2 Shall provide puzzles to aid the player in learning the morph functionality. (cbatten, priley, ddang)

3.3.6.3 Shall provide hints to aid the player in solving each puzzle. (cbatten, priley, ddang)

3.3.6.4 Shall provide tutorials to explain:

- The game loop
- Trigger functions
- Player-made functions
- How to reference objects in the scene (cbatten, priley, ddang)

Puzzle concept

3.4 Compiler Algorithm (Functional Requirement)

3.4.1 Compile

3.4.1.1 Shall provide a mechanism to allow runtime compilation of player-edited scripts (ndearce)

3.4.1.2 Shall alert the player of any compilation errors in player-edited scripts (ndearce)

3.4.1.3 Shall provide the player with an error message identifying compilation errors in player-edited scripts (ndearce)

3.4.2 Apply to object

3.4.2.1 Shall apply successfully compiled code to objects within the scene (ndearce)

3.5 Performance Requirements

3.5.1 What will it run on.

3.5.1.1 Shall be compatible with Windows. (ksantos)

3.5.1.2 Shall be compatible with Linux. (ksantos)

3.5.1.3 Shall be compatible with MacOS. (ksantos)

3.5.1.4 Shall be compatible with Intel i3 6th generation processor. (ksantos)

3.6 Assumptions

3.6.1 Team Silver shall assume that the user does not want to harm his own PC and therefore will not seek to find malicious code to implement in their game (mtuckson)

3.7 Non-Functional Requirements

3.7.1 Security

3.7.1.1 Shall provide the ability to scan player's entered code for using/include statements through a tool associated with Unity. This will aid in blocking the players from entering malicious code. (ceveritt)

3.7.2 Reliability

3.7.2.1 Shall ensure all levels are 100% completable using the methodology taught within the specific level, however levels may include alternate solutions. (jstokes)

3.7.2.2 Shall properly test levels to ensure a bug-free product, with a minimum of at least 75% bug free for the initial prototype (jstokes)

3.7.2.3 Critical Items (jstokes)

3.7.2.3.1 Shall allow the player to easily edit their code with a user-friendly code manipulation box, and displays which objects allow manipulation through the compiler UI. (jstokes)

3.7.2.3.2 Shall allow the player to manipulate and run the code on specific, selectable objects inside the level using the compiler algorithm.

3.7.2.3.3 Shall provide code correction tips (if desired by the player) for the current level’s taught technique. (jstokes)