Changes to Design Document and Discussion

Connor Sedwick, Behnam Saeedi, Collin Dorsett

✦

1 Changes to Definitions:

- **Abstraction**: redefined as a overarching term to describe the use of Blocks in the Scene/Build space.
- **Layers**: redefined as separate builds created by user that are created as standalone files to be used as Blocks.
- **Block**: Blocks are abstracted objects representing Python code syntax. These blocks could be classified as:
  - **Data**: This portion is responsible for data handling in the document.
  - **Methods**: This portion is responsible for function calls and definitions.
  - **Probes**: This portion is responsible for manipulating values between Data and Method Blocks.
  - **Class**: This portion is responsible for class definitions.

2 Discussing Changes to Design

Over the course of this project the team has worked diligently to ensure that our documentation matches well with our current design. Unfortunately, though plans were laid out in detail early in the Fall they were subject to a few notable changes. Originally, outlined in our design document, we had decided that the way layers would be implemented would be in a sense like an overlay for users to draw and abstract away part of their build. After researching multiple ways of implementing this and running into a few roadblocks we eventually decided that layering would need to be implemented in another manner. As such, layering now requires that the user has created a diagram in the build space. Upon the user clicking on the ″Make Layer″ button, the code from the diagram currently on the build space will be interpreted and placed in a numbered layer folder (i.e. Layer1.py). The user then must click the ″Delete″ button to clear the build space allowing them to create their next layer. The plan for layers is to add the ability for users to spawn their created layers as abstracted blocks and allow them to connect them in the build space for them to be interpreted.

Another notable change to our design is the idea of the Different types of data blocks. As variable blocks are currently implemented, they handle a great deal of the operations required such as providing boolean operators to statements as inputs as well as handle string a numerical values.

Regarding some of the buttons on our GUI, there have also been major changes to their underlying structure. For instance, no longer is there s ″Build″ button, but instead it has become synonymous with the ″Extract″ feature we have implemented and thus has been removed. We would also like to note that the Extract feature does not open up a window for users to choose the destination of their project, instead we have implemented a text input bar by the ″Layers″ button for users to type the path to the destination folder.

One of the more noticeable changes is that we do not support the use of differing shapes for blocks in our software. This decision was made after realizing that blocks would be difficult to add text entry fields and buttons to if they were awkward shapes. This led to the decision to identify block types as different through their depicted structure and labeling. To elaborate, blocks are now all the same shape and only differ by their offered features and some labels shown on them.

To conclude, many of our projects features still adhere to our design document, but some have been changed slightly and some features have been removed as a result. Many of the decisions made regarding these features were in order to facilitate the development of the software in the interest of time. As such, some features are yet to be implemented such as the ”Stop” and ”Run” buttons that allow users to execute their code. Though they are not implemented yet their underlying functionality appears with our Extract feature. Changes will continue to be made as further development is carried out into the next year at the request of our stakeholder, Professor Fuxin Li.