GUI Programming  
Module-Maker



Julian Pahor  
10th August 2022

**Purpose**

Module-Maker is a game centered around having multiple musical boxes (modules) combining to make some form of music. You have several different types of modules that do different things, and you can interact with them to change their parameters. It’s more of an audio playground rather than a game, where you can mess around and potentially create something pleasant.

**Requirements**

The program will be designed and built fully in Unity and won’t need to depend on any other dependencies or pre-installed software besides to run it. It should be low in terms of its demand for processing power.

**Features**

Describe features that are a requirement for your tool. Eg: drag / drop commands, switching levels, restart levels, run through actions etc.

* Command Key to interact with modules  
  Description: Having some functionality to be able to walk up to a module and press a key in order to interact with it and bring up its own separate UI.
* Sliders  
  Description: Lots of various sliders used to change parameters within the modules, as well as for master settings for the whole game. (E.g., Master volume)
* Command Key to spawn and place modules.  
  Description: The ability to press a command key to open a window with various module choices. Being able to allow the player to pick and then place whichever module they would like to use.
* First-person character controller  
  Description: The control of the player from a first-person perspective is necessary for the program to run as well.

* Pop-Up Tool tips for Controls  
  Description: When the player is carrying out an action there will be a tooltip that pops up to help instruct them on any necessary controls needed to carry out that action.

**UI Wireframe Mockups**

Default Screen Overlay:

|  |  |
| --- | --- |
|  | The overlay will always have a few helpful tooltips on the screen to remind/assist players with controls to interact with the program environment. However, they will be placed on the edges of the screen to avoid disturbing the view of the player. |

Default Screen Overlay: Events:

* Options Button:  
  This is the button situated in the top left corner of the screen. It will pop open the settings user interface onto the middle of the screen to allow the player to change the overall settings for the environment. (E.g., Master Volume)
* Controls for Module Maker:  
  This is the graphical display in the bottom left corner of the screen. It has no interaction function but simply displays the controls to the player for opening the Module interface.
* Interaction Pop-Up:  
  This would be another non-interactable display that pops up when the player is facing an interactable module and provides the key binding for interaction.

*Module-Maker*:

|  |  |
| --- | --- |
|  | This is the interface that pops up when the player presses the key binding to open the “*Module-Maker*”. It will also trigger the mouse to unlock and allow the player to click and select one of the buttons that have appeared.  A couple of elements from the default interface remain on the screen as well, being the options button and tooltip for the key binding to open + close this pop-up. |

*Module-Maker* Events:

* Module Buttons:  
  In total there are 4 Module buttons that live within this interface. In the wireframe they are represented by the hexagons. The player can mouse over one of the buttons and select which Module they would like to be spawned into the game world. After the button has been selected it will close this interface and turn off the mouse as well to allow the player to start moving naturally in the environment again.
* Exit Button:  
  This is the button in the top right corner of the *Module-Maker* interface. This is just a simple button used to close the window should the player decide that they did not want to spawn a module. Pressing the key-binding used to open the interface would do the same thing however this provides the user the options to use the mouse on screen if they want. After the interface is closed the mouse also disappears and returns the mouse look functionality to the player.

Oscillator Interface:

|  |  |
| --- | --- |
|  | This is the interface for the Oscillator Module object. It allows the user to change the main functionality of the oscillator. It provides the user with 3 sliders to change the value of the frequency of the oscillator with two of them, and the volume with the final one. These sliders will have their value ranges set within themselves to only allow for reasonable value changes. This is most important for the volume slider so that it highly reduces the chance of the player hurting themselves via loud audio.  It is also important to note that this interface is diegetic and exists in the game world so the player will walk up to the module object, within the game environment, and interact with it to pull up this interface. |

*Module-Maker* Events:

* Freq Slider:  
  The frequency slider will directly change the rate at which the oscillator is oscillating (changing the pitch of the type of wave it is producing). Having a high range such as 30Hz – 6000Hz for example.
* Fine Tune Slider:  
  The fine tune slider also influences the frequency value of the oscillator. However, it is of a much smaller scale so that the user can make fine adjustments and get the frequency to the exact value that they want.
* Volume Slider:  
  The volume slider impacts the gain value of the oscillator which changes the amplitude of the wave it is producing. Changing the volume that this instance of an oscillator is outputting.

**Testing**

During the process of developing the program it was tested by another classmate and the following was observed during testing:

Gameplay Observations:

-Player immediately tried to sequence break past accepting the disclaimer

-Player immediately spawned a module below the floor where they would not have been able to interact with it

-Occurring multiple times, the player would try to interact with the module without using the mouse and pointing the player at the corresponding buttons/sliders they were trying to interact with.

-Modules were being used to construct some form of a jumping puzzle with the player continuing the build up into the sky

-Module spawn positions were heavily incorrect when trying to spawn them while standing on another module (To the point where you could not see the wireframe mockup during placement)

-Player automatically knew that the F key would close the “*Module-Maker*” menu after opening it with it. (They never used the mouse to hit the close button in the top right of the module menu)

-Jumping seemed very unresponsive when player model was on top of the modules.

Verbal feedback after playtest session:

-Interacting with a module could put the player into a locked state and bring up the mouse automatically / use some form of crosshair in the middle of the screen to allow the player to interact with the diegetic UI.

-The use of the mouse itself felt awkward when interacting with the world space UI but even more so the key binding (‘P’) forced the player to look down at the keyboard and bring their hand off the mouse to bring up the mouse.

Potential fixes to consider:

-Easiest fix for the Mouse would be to change the key binding, potentially to the ‘F’ key to allow the player to toggle it with their left hand.

-Ideally the addition of a crosshair in the middle of the screen to allow players to interact with the UI more naturally, instead of having to bring up the mouse. (To further eliminate the use of the on-screen mouse the disclaimer could be accepted using ‘Enter’ and the module maker could make use of numbers ‘1-6’ to select which module the player would like to spawn.)

-Fix ground checks to allow players to easily jump on modules to allow them to build up into the sky, this would also investigate fixing the module placement bug that was occurring when placing modules while standing on one.