|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Input | SFML::Input | Input | Event | Notice |
|  |  |  |  |  |
| Space bar | Sf::KeyPress | SpaceBarInput | SimPauseEvent | SimPauseNotice |
| Speed Up Button |  | SpeedUpInput | SimSpeedUp | SimSpeedNotice |
| Slow Down Button |  | SlowDownInput | SimSlowDown | SimSlowNotice |
|  |  |  |  | SimChangeNotice |
|  |  |  |  |  |
| Select a Node |  | SelectNodeInput | SendNodeData | NodeDataNotice |
|  |  |  |  |  |
|  |  |  |  |  |

--------------------------------------------------------------------------------------------------------------------------

Quick rundown of MVC system

sf::Event -> Viewer -> Input -> Controller -> Event -> Model -> Notice -> Viewer

Viewer gets SFML::Input, creates an Input based on SFML::Input, sends Input to Controller, Controller creates an Event based on the received Input, Controller sends Event to Model, Model acts based on the received Event, and Model sends a Notice to the Viewer about what’s changed.

--------------------------------------------------------------------------------------------------------------------------

General requirements of Input, Event, and Notice derivatives

All derivatives of Input, Event, and Notice classes must have a defined InputType, EventType, or NoticeType with a name describing what class that the type is associated with. Enumerations of types can be found in Input.h, or Event.h, or Notice.h.

Additionally the constructor of the derivative class MUST set the type attribute to the appropriate Type.

Example: SpaceBarInput has an InputType called SPACEBAR and a constructor that sets type = InputType::SPACEBAR

---------------------------------------------------------------------------------------------------------------------------

MVC Methods:

Controller.parseInput() – Controller iterates through each Input in the queue. In this method you can specify what Event type should be created according to the current Input type, and then store the new Event in the queue to be sent to the Model component.

Model.update() – Model iterates through each Event in its queue and performs operations based on what Event type is currently being processed.

Viewer has no predefined method for handling SFML::Inputs. Generally, the method would poll the Window for an sf::Event, determine what inputs occurred based on what data is contained in the sf::Event, and then create an appropriate type of Input object and store it in the Input queue to be sent to the Controller.

Viewer.processNotice(Notice\* n) – Viewer can perform operations based on what type of Notice is currently being pointed to by n.

Viewer.updateDisplay(Notice\* n) – Viewer can use the type of Notice pointed to by n to determine what components of the display need to be updated for this Notice, and get needed data from the Notice instance.