**Assignment 1 Architecture**

Cory Smith

**Unit Manager**

enum UnitType - The different types of units

Variables:

* map<UnitType, map<int, KinematicUnit\*>\*> mMapList – A map that uses the UnitType enum for its keys. Its value is another map of the units in the scene that are accessed through their ID numbers
* stack<int> mAvailableIDs – Stack of ID’s that can be assigned to units. When units are removed their id is pushed into the list
* IDType mPlayerIconBufferID & mEnemyIconBufferID– ID numbers for accessing the player and enemy sprite buffers

Functions:

* Sprite\* getUnitSprite(UnitType \_unitType) – Returns the sprite for the passed in unit type
* map<UnitType, map<int, KinematicUnit\*>\*>\* getMapList() – Returns mMapList
* map<int, KinematicUnit\*>\* getUnitMap(UnitType \_type) – Returns mMapList at \_type
* KinematicUnit\* getUnit(int \_ID, UnitType \_type) – Returns the unit specified by their ID & type
* void update(float \_dt)– Calls update for all units in the scene
* void draw(GraphicsBuffer\* \_buffer) – Calls draw for all units in the scene
* KinematicUnit\* addUnit(UnitType \_type, (KinematicUnit constructor params)) – Creates a unit based on the \_type adds it to the map and returns it
* bool removeUnit(int \_ID) – Removes the unit specified by the ID
* void removeRandomUnit()– Removes a random unit from the map omitting the player. If the map only has the player afterwards it ends the game

**Input System**

Variables:

* map<int, bool> mkeyStates – A map of Booleans that uses allegro key codes as its keys. Used to see if a key was being pressed last frame or not
* bool mMouseDown; – Similar to the key states map, just a single Boolean for the mouse button

Functions:

* void fireEvent(InputType \_type, int \_keyCode)– Sends a new inputmessage to the message manager based on the key that was pressed and the type of key press it was
* void update() – Checks the state for all inputs and fires events based on key presses
* Vector2D getMousePosition() – Returns the current mouse position

**Input Message**

enum InputType – Enumerates how the key input was given

Variables:

* InputType mInputType – The type of input this message is for
* int mKeyCode – The key that was pressed

Functions:

* InputType getType() – Returns mInputType
* int getKeyCode() – Returns mKeyCode
* void process() – Processes the message
* void addUnitByType(UnitType \_type) – Tells unitmanager new unit based on the unit type
* void deleteRandomUnit() – Calls unitmanager.deleteRandomUnit()