**Greed Designer**

1. How did you apply inheritance to your program's design?

I used diagram of classes and abstract to understand what classes is super and subclasses.

1. Identify the objects in your program.

* *actor*
* *artifact*
* *cast*
* *director*
* *keyboard*
* *video\_service*
* *point*
* *color*
* *score*

1. Define the responsibility, behaviors and state for each object.

|  |  |  |  |
| --- | --- | --- | --- |
| Object | Responsibility | Behavior’s | State |
| actor | The responsibility of Actor is to keep track of its appearance, position and velocity in 2d space | \_text = String  \_font\_size = number  - To instance color and Point | \_text  \_font\_size  \_color  \_position  \_velocity |
| artifact | The responsibility of an Artifact is to provide a message about itself | Superclasse(Actor)  \_message: String | get\_message  Set\_message |
| cast | The responsibility of a cast is to keep track of a collection of actors. It has methods for adding, removing and getting them by a group name. | \_actors: dicionary | \_add\_actor  \_get\_actor  \_get\_all\_actor  \_get\_first\_actor  remove\_actor |
| keyboard | The responsibility of a KeyboardService is to detect player key presses and translate them into a point representing a directio | \_cell\_size: cell\_size | get\_direction |
| Director | To control the sequency of play | -To instance keyboard, cast, video\_service, | startgame  \_get\_inputs  \_do\_updates  \_do\_outputs |
| Video\_Service | The responsibility of the class of objects is to draw the game state on the screen | Get parameters from main.py as  \_caption: number  \_width : number \_height: number  \_cell\_size: number  \_frame\_rate: number  \_debug: number | close\_window  clear\_buffer  draw\_color  draw\_actors  flush\_buffer  get\_height  get\_width  is\_window\_open  open\_window  \_draw\_grid |
| Color | The responsibility of Color is to hold and provide information about itself. Color has a few convenience methods for comparing them and converting to a tuple | Get parameters from main.py as  \_red: number  \_green: number \_blue: number \_alpha: number | To\_tuple |
| Point | The responsibility of Point is to hold and provide information about itself. Point has a few convenience methods for adding, scaling, and comparing them | Get parameters from main.py as  \_x: number  \_y: number | add  equals  get\_x  get\_y  scale |
| Score | The responsibility of Point is to hold and provide information about itself. Score has a few convenience methods for adding and comparing them |  |  |

1. Identify the relationships between your objects.

Director > KeyboardServices

> video\_service>

> Actor>Artifact>

> Actor>Cast>

> Actor>Color>Point

1. Translate your object designs to class designs.

*Diagram below*

Document your objects and classes so that everyone can refer to them throughout the project. The graphical notation that follows is often the easiest way. It really doesn't matter how you do it though. It just matters that you do.

*Diagram below*

