**Cycle Design Planning**

We will be implementing the principle of polymorphism together with other principles we’ve learned to create the Cycle game. There shall be at least one method that will be overridden as part of the program.

The program will be made up of the below classes and methods.

**Class: Actor (will be the parent class)**

* Will be visible and movable in the game, and track its appearance, position and velocity
  + Attributes:
    - \_text: To display text
    - \_font\_size: text to have a color
    - \_position: screen coordinates
    - \_velocity: speed and direction

**Class: Cast**

* Will keep track of all the actors. It’ll have methods for adding, removing and grouping actors.
  + Attributes:
    - \_actors: dictionary of actors {Key: group\_name, Value: list of actors}

**Class: Score**

* A record of points made or lost. Keep track of points scored by each player. (Maybe we should include food?)

**Class: Snake1**

* Will be responsible for moving the first snake.
  + Attributes:
    - \_points: number of points the food is worth

**Class: Snake2**

* Will be responsible for moving the second snake
  + Attributes:
    - \_points: number of points the food is worth

**Class: Director**

* Will control the sequence of play
  + Attributes:
    - \_video\_service: provides video output

**Class: Action**

* Responsible for doing something that is integral or important in the game. It has one method, execute(), which should be overridden by derived classes.

**Class: ControlActorsAction (Child to Action class)**

* Input action that controls the snake. It gets directions and moves the snake’s head.
* An example of method overriding
  + Attributes:
    - \_keyboard\_service: an instance of KeyboardService

**Class: DrawActorsAction (Child to Action class)**

* Output action that draws all the actors. It draws all the actors.
* An example of method overriding
  + Attributes:
    - \_video\_service: an instance of VideoService

**Class: HandleCollisionsAction (Child to Action class)**

* An update action that handles interactions between the actors. It handles the situation when the snake collides with itself or another snake.
* An example of method overriding
  + Attributes:
    - \_is\_game\_over: whether or not the game is over.

**Class: MoveActorsAction (Child to Action class)**

* An update action that moves all the actors. It moves all the actors that have a velocity greater than zero.
* An example of method overriding

**Class: Script**

* It keeps track of a collection of actions. Has methods for adding, removing, and grouping them.
  + Attributes:
    - \_actions: a dictionary of actions {Key: group\_name, Value: list of actions}

**Class: KeyboardService**

* Will detect player key presses and translate them into a point representing a direction.
  + Attributes:
    - \_keys: letters to key mapping

**Class: VideoService**

* Will draw the game state on the screen

**Class: Color**

* Will hold and provide information about itself
* Has a few convenience methods for comparing them and converting to a tuple.
  + Attributes:
    - \_red: red value
    - \_green: green value
    - \_blue: blue value
    - \_alpha: alpha value

**Class: Point**

* Will hold and provide information about itself.
* Will have methods for adding, scaling, and comparing.
* Origin (0, 0)
  + Attributes:
    - \_x: horizontal distance from origin
    - \_y: vertical distance from origin