Cicle Designer

How did you apply polymorphism in your program's design?

I will using on my project the same class on the cicle project and create another snack, using polymorphism to create a new behavior to the second cicle.

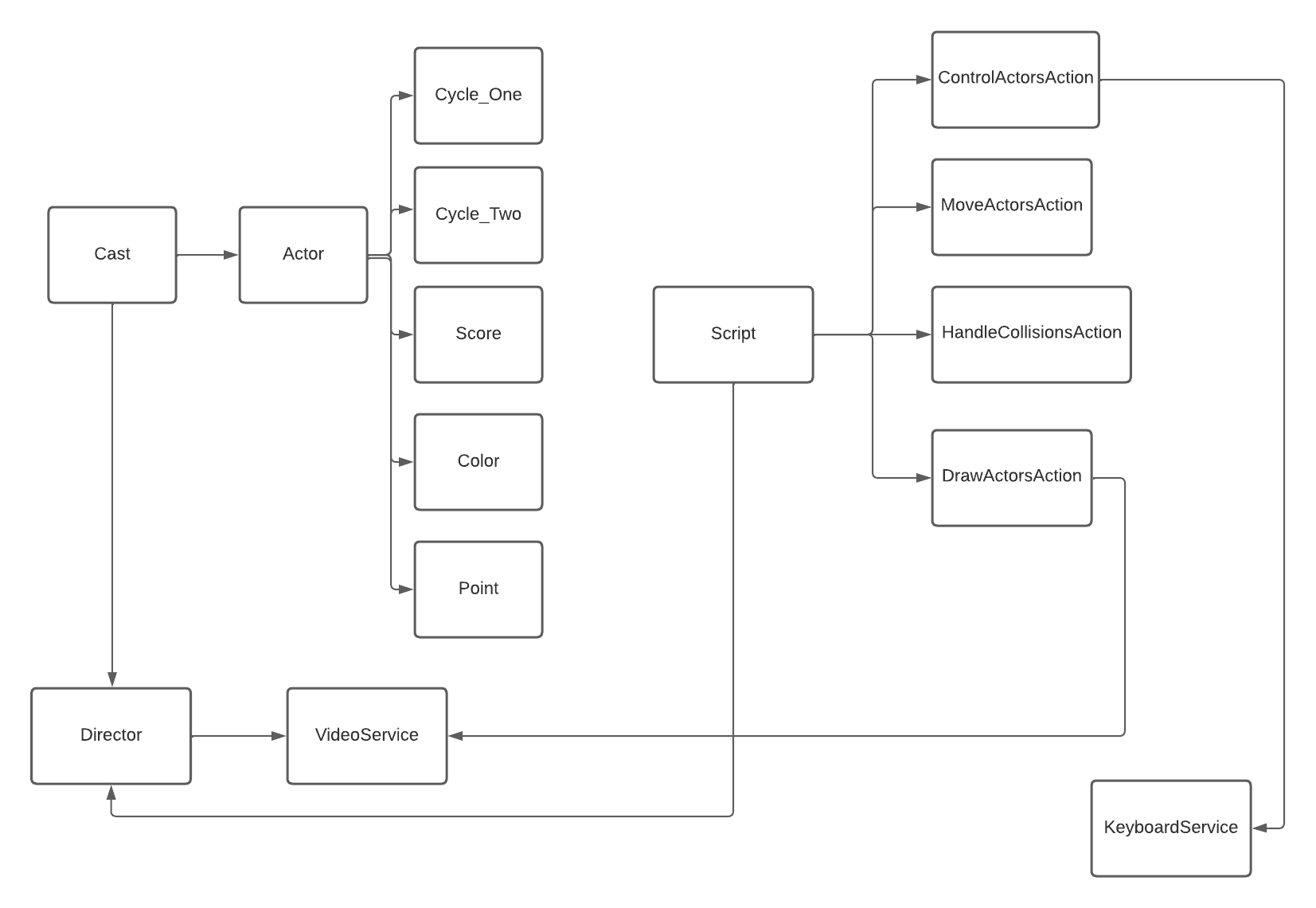
1. Identify the objects in your program.

|  |  |  |
| --- | --- | --- |
| * Actor | * Director | * Script |
| * Cast | * Action | * Keyboard\_service |
| * Score | * Control\_actors\_action | * Video\_service |
| * Cycle\_One | * Draw\_actors\_action |  |
| * Cycle\_Two | * Move\_actors\_action |  |

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

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Class | Responsibility | Behavior’s | State |
| Actor | The responsibility of Actor is to keep track of its appearance, position and velocity in 2d space | - Gets the actor's color as a tuple of three ints (r, g, b).  - Gets the actor's font size  - Gets the actor's position in 2d space  - Gets the actor's textual representation  - Gets the actor's speed and direction  - Moves the actor to its next position according to its velocity. Will wrap the position from one side of the screen to the other when it reaches the given maximum x and y values.  - Updates the color to the given one.  - Updates the position to the given one  - Updates the font size to the given one  - Updates the text to the given value  - Updates the velocity to the given one | -get\_color  -get\_font\_size  -get\_position  -get\_text  -get\_velocity  -move\_next  -set\_color  -set\_position  -set\_font\_size  -set\_text  -set\_velocity |
| 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 | - Adds an actor to the given group  - Gets the actors in the given group  - Gets all of the actors in the cast  - Gets the first actor in the given group  - Removes an actor from the given group | - add\_actors  - get\_actors  - get\_all\_actors  - get\_first\_actor  - remove\_actor |
| Score | The responsibility of Score is to keep track of the points the player has earned. It contains methods for adding and getting points. Client should use get\_text() to get a string representation of the points earned. | - Adds the given points to the score's total points | - add\_points |
| Cycle\_one | The responsibility of Cicle is to move itself | - get segments the cicle  - loop each segment and moves the segment to its next position according to its velocity  - get position on the head segments  - Add new segment to the grow tail of cicle  - set velocity to turn the cicle head  - To create the body to cicle and add segments | - get\_segments  - move\_next  - get\_head  - grow\_tail  - turn\_head  - \_prepare\_body |
| Cycle\_Two | The responsibility of Cicle is to move itself | - get segments the cicle  - loop each segment and moves the segment to its next position according to its velocity  - get position on the head segments  - Add new segment to the grow tail of cicle  - set velocity to turn the cicle head  - To create the body to cicle and add segments | - get\_segments  - move\_next  - get\_head  - grow\_tail  - turn\_head  - \_prepare\_body |
| Director | The responsibility of a Director is to control the sequence of play | - Starts the game using the given cast and script. Runs the main game loop.  - Calls execute for each action in the given group | - start\_game  - execute\_actions |
| Action | The responsibility of action is to do something that is integral or important in the game. Thus it has one method, execute(), which should be overridden by derived class | - Executes something that is important in the game. This method should be overridden by derived classes | - execute |
| Control\_actors\_action | The responsibility of ControlActorsAction is to get the direction and move the cicle's head | - Executes the control actors action | - execute |
| Draw\_actors\_action | The responsibility of DrawActorsAction is to draw all the actors. | - Executes the draw actors action | - execute |
| HandleCollissionsAction | The responsibility of HandleCollisionsAction is to handle the situation when the cicle collides with segments, or the game is over | - Executes the handle collisions action  - Sets the game over flag if the cicle collides with one of its segments  - Shows the 'game over' message and turns the cicle if the game is over | - execute  - \_handle\_segment\_collision  - \_handle\_game\_over |
| Move\_actors\_action | The responsibility of MoveActorsAction is to move all the actors that have a velocity greater than zero. | - Executes the move actors action | - execute |
| Script | The responsibility of Script is to keep track of a collection of actions. It has methods for adding, removing and getting them by a group name | - Adds an action to the given group  - Gets the actions in the given group.  - Removes an action from the given group | - add\_action  - get\_actions  - remove\_action |
| Keyboard\_service | The responsibility of a KeyboardService is to indicate whether or not a key is up or down | - Checks if the given key is currently up  - Checks if the given key is currently down | - is\_key\_up  - is\_key\_down |
| Video\_service | Outputs the game state. The responsibility of the class of objects is to draw the game state on the screen | - Closes the window and releases all computing resources.  - Clears the buffer in preparation for the next rendering. This method should be called at the beginning of the game's output phase.  - Draws the given actor's text on the screen.  - Draws the text for the given list of actors on the screen.  - Copies the buffer contents to the screen. This method should be called at the end of the game's output phase.  - Whether or not the window was closed by the user  - Opens a new window with the provided title  - Draws a grid on the screen  - Get the measure of text string | - close\_window  - clear\_buffer  - draw\_actor  - draw\_actors  - flush\_buffer  - is\_window\_open  - open\_window  - \_draw\_grid  - \_get\_x\_offset |
| 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 | - Gets the color as a tuple of four values (red, green, blue, alpha). | - 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. | - Gets a new point that is the sum of this and the given one.  - Whether or not this Point is equal to the given one.  - Gets the horizontal distance.  - Gets the vertical distance.  - Reverses the point by inverting both x and y values.  - Scales the point by the provided factor. | - add  - equals  - get\_x  - get\_Y  - reverse  - scale |

1. Identify the relationships between your objects.



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*

