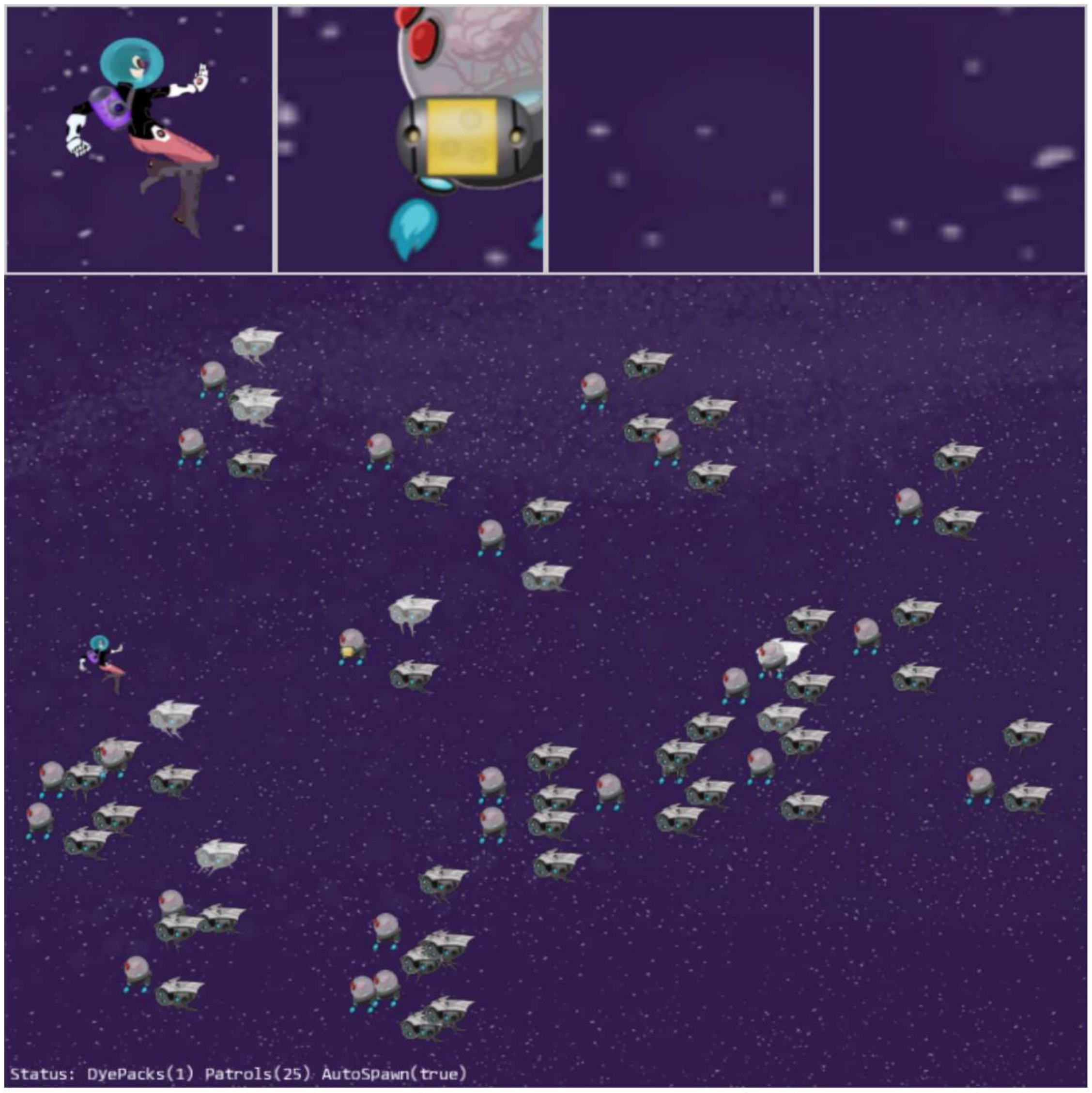
CSS452: Programming Assignment #5
Behaviors, Bounds, and Interpolation

Due time: Please refer to our course web-site

Objective
In this programming assignment we will experience with defining simple behaviors, understand and extend bounding boxes, work with controls governed by equations, and camera manipulations.

Assignment Specification:
Here is an example of the results from this assignment:



Status: DyePacks(1) Patrols(25) AutoSpawn(true)

In the above screen shot, three types of objects, and five camera views. In my implementation, all objects are defined based on the *minion_sprite.png* sprite sheet we have been working with in class examples. The followings are some the detail.

<u>Your Canvas</u> and <u>World:</u>

- Define a reasonable canvas resolution (e.g., mine is 800x800)
- Your world must be at least 200-unit in size (e.g., mine is 200x150)
- The viewports (details later): in my case
  - Large Viewport: 800x600
  - Small: 200x200 (there are four of these at the top row)

<u>The Objects</u>:

- **Hero**: This is the Dye character.
  - **Size**: 9x12
  - **Behaviors**:
    - **Motion**: When the mouse is inside the large viewport, Dye will always try to move toward the mouse position with interpolation values of:
      - **Rate**=0.05
      - **Cycles**=120 frames
    - **Hit:** When the Hero bound collides with the bound of the **Head** of a **Patrol** (define later) when a Hero is hit, its size oscillates according to our damped harmonic function where:
      - **X/Y Amplitude:** 4.5, 6 (50% of the size)
      - **Frequency**: 4
      - **Duration**: 60 frames
  - **User input**:
    - **Space bar**: each time when the space bar is hit, the Hero spews out a DyePack. Refer to the following for details.
    - **Q**: For testing (and grading) purposes: the Q-key click triggers a Hero hit event.

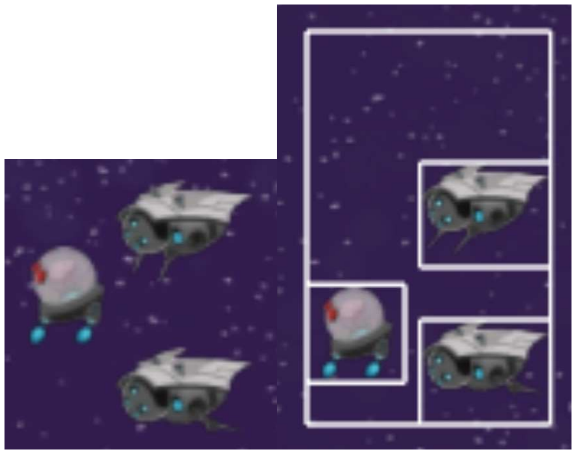
- **DyePack**: This is the DyePack. Always spawned at the Hero's location (e.g., from her hand).

  - **Size**: 2x3.25
  - **Behaviors:**
    - **Motion:** Travel at a constant speed in the positive-X direction.
      - **Speed:** 120 units per second
      - **Slows down:** if encounters the bounds of a Patrol, speed deaccelerates at a constant of 0.1-unit/frame (**NOTE:** this is a decrease by a constant number, ***not a constant rate***)
    - **Hit**: When pixel collide with a **Patrol**. A DyePack will oscillates according to our damped harmonic function where:
      - **X/Y Amplitude:** 4, 0.2
      - **Frequency**: 20
      - **Duration**: 300 frames [we can grade this behavior]
    - **Lifespan:**
      - **Maximum** lifespan of 5 seconds.
      - **Terminates** if
        - travels outside of the World Bound
        - speed reaches 0 (or less)
        - after Hit (hitting a **Patrol** (details later))
  - **User Input:**
    - **D**: D-Key *pressed* (NOTE: this is key pressed) triggers slow down
    - **S**: S-Key click triggers a Hit event for *ALL* DyePack currently on in the world

- **Patrol**: This is an object consisting of three separate textures/objects: a **Head** and two **Wings**.
  - **Head**: This is the "leader" of the Patrol.
    - **Size**: 7.5x7.5
  - **Wing**: The two Wings objects are
    - **Size**: 10x8

        - **Locations**: both of the Wing object is *tries* to maintain to be located at 10 units to the right of the Head, where the Top Wing is about 6 units above and the Bottom Wing is about 6 units below the Head's y position.
        - **Color**: initialize all object color to [1, 1, 1, 0], note, color[3] is 0.
    - **Behaviors:**
        - **Head Motion:** The Patrol Head travels in random directions with a speed random between 5 to 10 units/second.
        - **Wing Motion:** The Wings interpolate from their current positions in an attempt to follow the Head:
            - Rate: 0.05
            - Cycles: 120 frames
        - **Animation:** at least one of the Wing objects must have a simple sprite animation defined
        - **Spawning:**
            - **Auto Spawn Rate:** When enabled, a new Patrol is spawned randomly between 2 to 3 seconds.
            - **Spawn location:** right-half of the world, bounded between top/bottom 25%
        - **Bound and Movement:**
            - **Bound:** The size of the control bound is defined as:
                - **Width:** The combined width of the bounds of Head and Wing
                - **Height:** 150% of the combined height of the bounds of Head and Wing

              Here are a couple of close ups on the Patrol and a Patrol with bounds switched on.



            - **Movement:** When the bound of the entire bound touches the bounds of the WC, the direction of the velocity reflects.
        - **Hit:** When a DyePack collides with
            - **Head:** The position of the Head (only) is moved towards the right by 5 units.
            - **Wing:** The alpha channel, color[3], of the colliding Wing is incremented by 0.2
        - **Lifespan:** terminates if
            - The entire bound is to the right of the WC window (can occur if being Hit on the Head continuously)
            - The Alpha channel of either of the Wing becomes equal or larger than 1.0
    - **User Input**:
        - **P**: P-key click, toggles auto spawning on/off
        - **C**: C-key click, spawns a new patrol
        - **B**: B-key click, toggles the drawing of all bounds for all the Patrols. Note, you can draw using the LineRenderable, and, make sure to draw both the bounds for individual members' as well as the entire Patrol object (as indicated by the above screen shot).
        - **J**: J-key click, triggers the Head Patrol hit event for ALL Patrol objects in the world.

Note: You are free (and encouraged) to use your own and/or different textures. Please do ensure that, for testing/grading purposes, your system has similar relative size and movements of the objects.

<u>The Camera Views</u>:

There are five camera views in the world:

- **Main Camera**: This is the camera associated with the bottom large Viewport. This camera is stationary and shows the entire world.

- **ZoomCams**: The top row are four zoom cameras.
    - A ZoomCams only draws when it is active: with either of the following conditions is true:
        - User forces the drawing of the ZoomCam (with numbers 0 to 3)
        - A Hit event occurs on the zoom target
    - The ZoomCam focuses on two of the object types:
        - The left-most ZoomCam: zoom subject is the always centered on the **Hero**
            - WC Width=15
        - The right-three ZoomCam: zoom subject is the centered of **DyePack** when it is oscillating
            - WC Width=6
    - When a DyePack Hit event occurs, the next inactive ZoomCam is found to draw the DyePack's motion. If all ZoomCams are active, the event is not drawn. A ZoomCam becomes inactive again after the DyePack it focuses on terminates.

<u>Status Message:</u>

At all point, your system must report:

- Number of Patrol units spawned
- Number of DyePacks spawned
- The state of Auto Spawn mode (on or off)

Hints:

1. My implementation is based on book Example-7.6 (For Line Support, on drawing the bounds).

2. This assignment is much simpler (in concept) than the previous one. However, it involves a little more work, where the challenge is adding simple behavior without overwhelming the complexity of the entire system. This means, planning and good design are important. Also, you should start early (as there are probably more coding).

3. In my implementation, I have the following classes defined:
    a. Hero
    b. DyePack (DyePackSet is basically GameObjectSet)
    c. ZoomCam, ZoomCamSystems
    d. Patrol, PatrolSet

    The important point to note is I have *explicit* group objects to handle the collect of objects as a set.

4. The requirement on supporting user input to generate events, like the Hero-Hit, DyePack hit, etc. Those are specified to give you hints on how to develop large systems: testing independently from the actual system.

5. Here is my approach in implementing something like this:
    a. First implement the DyePack. This is the simplest.
        i. SpaceBar spawns a DyePack at the current mouse position
        ii. Implement the slow down, hit behaviors
        iii. Test the behaviors with the defined Keys
        iv. Now, support generating many DyePack.
        v. Now, DyePack is working. Put it aside.
    b. Implement the Hero

        i. Implement the Hit behavior
        ii. Test this behavior with the Q-key
        iii. Now Hero is working, put it aside.
    c. Implement the Patrol by first with only the Head
        i. Test Head's hit behavior
        ii. Add in one Wing
        iii. Implement the interpolation of Wing towards Head.
        iv. Now, implement the second Wing element.
    d. Now, and only now, implement the interaction between the different objects
    e. Lastly, implement the ZoomCam

6. Key to the above implementation strategy?
    a. Make sure one object is working (almost) perfectly before moving on to the next
    b. DO NOT implement the interaction of objects UNTIL the objects themselves are done.

7. Camera movements, make sure you call **Camera.update()** somewhere in your updates(), otherwise, the camera will NOT move!!

Please do start early, this assignment is quite a bit of work!!

Creativity and Extra Credits: The App serves as a testing ground for the other objects. Do anything you want with your App (make sure to support all features being tested), and you can do anything you want! ☺.

Credit Distribution

Here is how the credits are distributed in this assignment:

1.	Hero object		20%
	a. Size, Interpolated Motion	10%	
	b. Support Hit behavior	10%	
	c. Support DyePack Spawning	10%	
	d. Support Q-Key	10%	
2.	DyePack		25%
	a. Size, Motion	5%	
	b. Slowdown when inside Patrol bound	10%	
	c. Support Hit behavior	10%	
	d. Lifespan: Timed, WC Bound, Speed, Hit	10%	
	e. Support: S, D Keys	10%	
3.	Patrol		35%
	a. Size, Bounded movement, Spawn	5%	
	rate/locations	5%	
	b. Wing animation	10%	
	c. Wings interpolating towards Head	10%	
	d. Support drawing of bounds	5%	
	e. Proper termination	10%	
	f. Support Head/Wing Hit	10%	
	g. Support: P, C, B, J Keys		
	Four-person team: Support two types of Patrol objects with behaviors similar to the above. Spawning 50% of the time. (15%)		
4.	Camera Views		10%
	a. Main Viewport	10%	
	b. 4x Zoom Views	10%	
	c. Proper on/off according to Hit events	10%	
	d. Support: 0, 1, 2, 3 Keys	10%	
5.	Status output		5%
	a. Number of active DyePack	5%	
	b. Number of active Patrol	5%	
	c. Patrol auto-spawn on/off	5%	
6.	Proper submission		5%
	a. Zip file names with **NO SPACES**	5%	
	b. No extra unused files/folders (E.g., Test folder)	5%	
	c. Styles (project name, variable names, etc.)	5%	

This programming assignment will count 20% towards your final grade for this class.