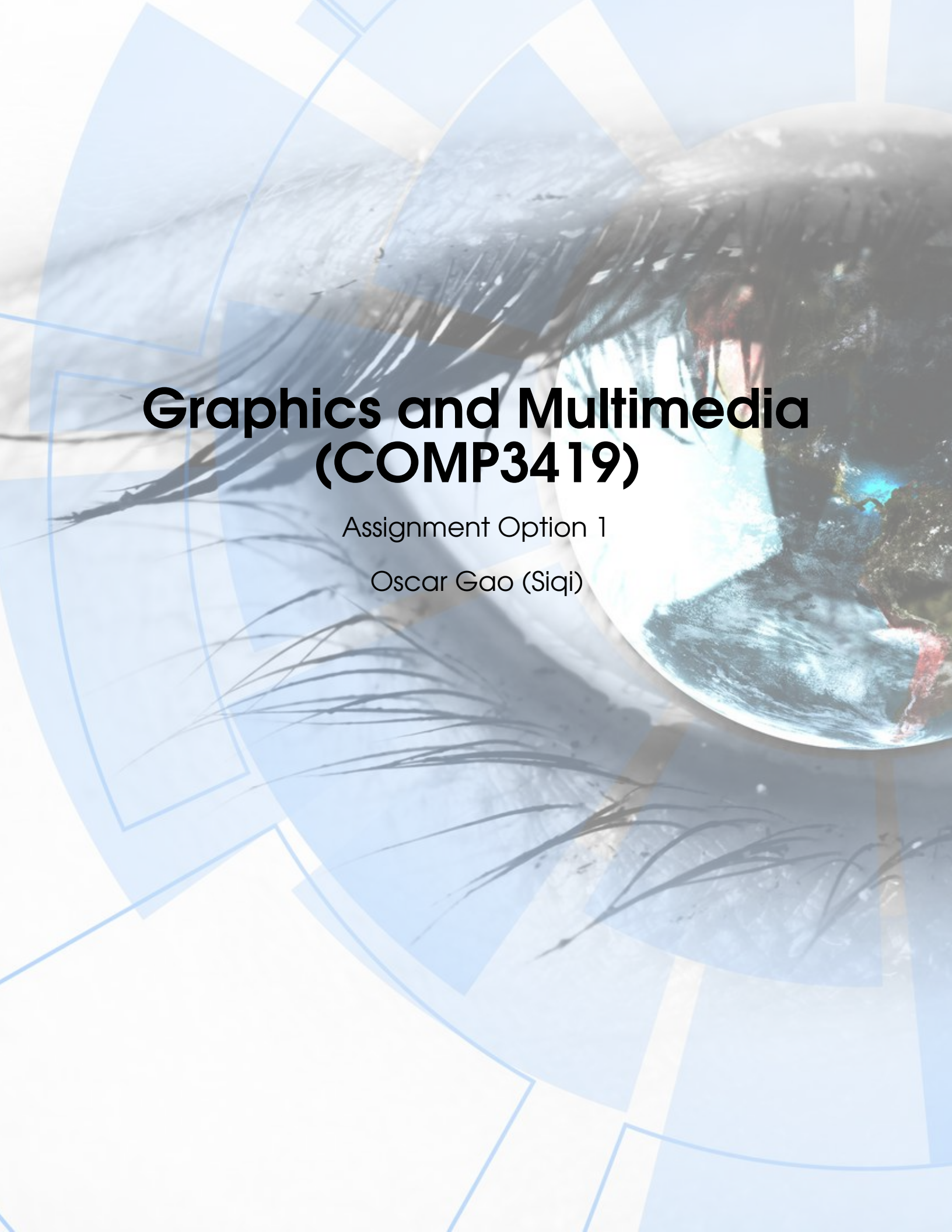
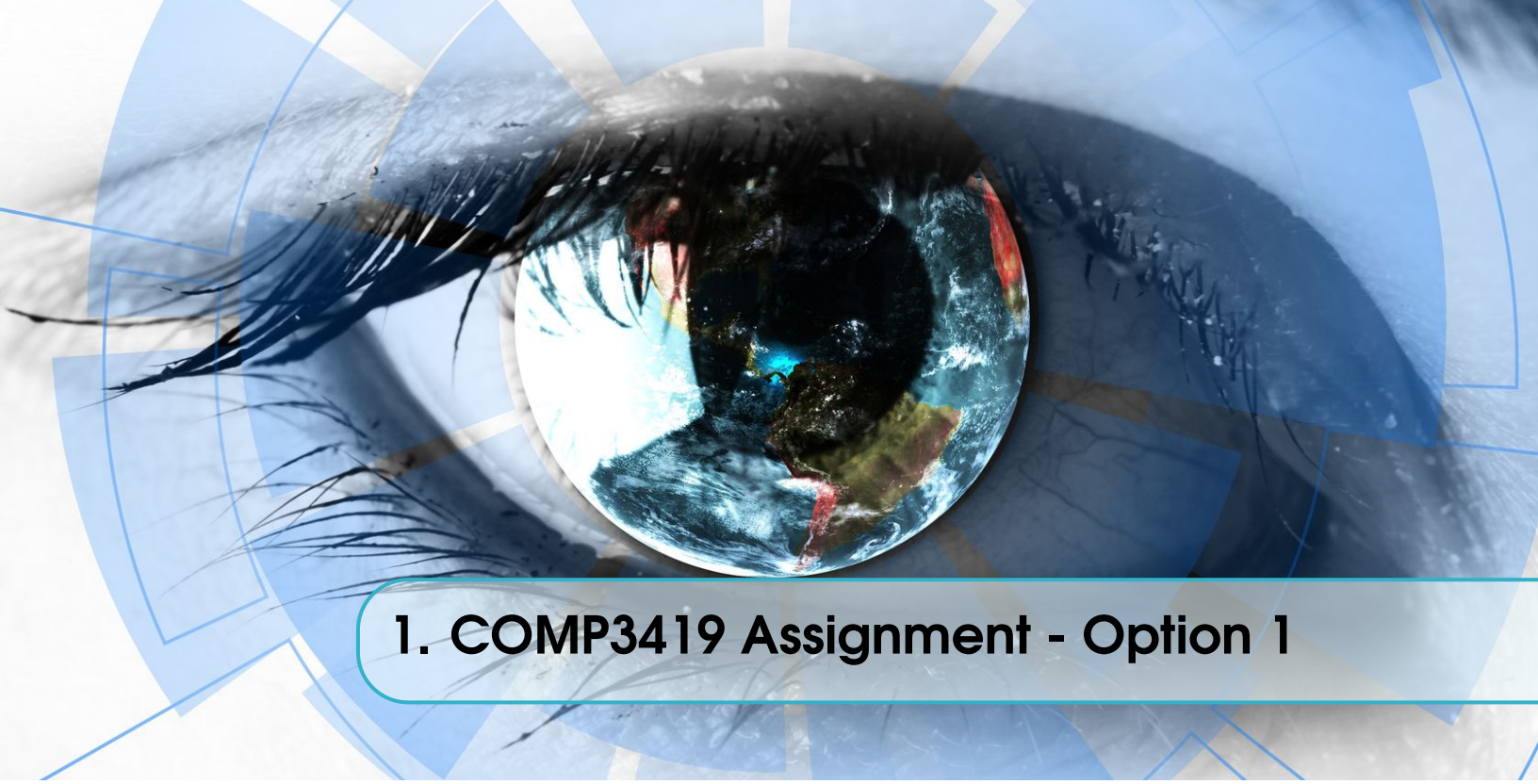


Graphics and Multimedia (COMP3419)

Assignment Option 1

Oscar Gao (Siqi)



1. COMP3419 Assignment - Option 1

1.1 Introduction

This is a Option 1 Report, where a marionette that moves according to the given video interacts with 2 objects, at 30FPS.

1.2 Story Development and Scriptwriting

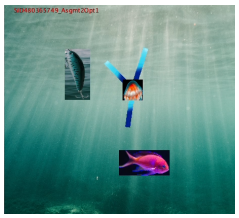


Figure 1.1: The scene

The scene is underwater, where a jellyfish, a fish, and the bait on a fishing rod can be seen infront of a backdrop of the water. The scene is during the day.

The main character is a jellyfish, with 2 arms and 2 legs differentiated by the color. The arms are dark blue from the torso to light blue at the hands. The feet are the opposite, light blue from the torso, changing to dark blue at the feet.

The two side characters are the fish and the bait.

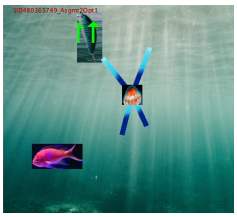
The fish moves randomly, changing the direction slightly so it often jitters. It cannot move past the walls, though part of it can go offscreen (though not fully).
 The bait moves up and down randomly, similar to the fish. However, it moves left and right in the direction of the fish.
 The events are as follows :

(a) Fish and rod moving

(b) Notice the change in direction

Figure 1.2: Movement as normal

(a) Fish being surprised by jellyfish



(b) Rod bouncingUp

Figure 1.3: Event 1

The main character hitting the side character will cause it to move. The fish moves to the left, and has arrow signs representing surprise. The rod will bounce up, with arrows representing its direction of movement).

The two side characters overlapping by a noticeable amount represents the fish getting caught on the bait, so the two move up irrespective of the main character, and leave the screen from the top. When that occurs, "time passes", and the background becomes night time. After 1 second, the fish and rod come back and the jellyfish returns to the start of its movement to represent a new day.

1.3 Motion Capture

I started by grouping all the red points (with gaps every 3 units in both directions) into groups based on the first point making a group, and then, adding all surrounding pixels into its group, and into the queue, and removing then from the list of points

Until the queue is empty, take the top point:

- Move points in range from red points into queue, and the group
- If no queue, take the first value from the list of red points, and make it a new group.

This way, I will have a list of red clusters.

I then averaged the values to get the average position, and the number of clusters.

I sorted them based on the amount of clusters, and got the top 5 values into the top5.txt file.

Reading from that file, I set the first frame's head, hands and limbs to the right values, and iterated through based on the smallest change from previous frame. If it had 5 units, then I would base it on the position of the top 5 (positionally) clusters as the hands and torso in the middle, with the bottom 2 as feet. Missing values were taken to be the previous, or the corresponding limb (hands

(a) Example of captured fish being dragged up

(b) Another instance where it is being dragged up

(c) Night time

Figure 1.4: Side character events.

get other hand, feet get other feet).

This information (position of each body part in every frame)was printed into a text file for processing called "position.txt"

1.4 Background and Marionette and Intelligent Objects and Events (Processing)

In processing, I had the background set to different images based on the states of my objects. This will be mentioned in more detail in Events, but it goes from a normal underwater image to a nighttime underwater image after the fish is pulled up, then goes back after 1 second passes.

The marionette is a jellygfish.

Body describes the jellyfish. I has PImage variables that represent the images of its hands, arms, and torso. An integer, counter, stores what stage it has progressed through the file describing Its position is a PVector. An arraylist of arraylists called frames stores the frames from the text file.

This is done via the constructor requiring an arraylist of arraylists as an input, which is stored as frames. points is taken to be the first frame (i.e. an arraylist of PVectors)

The iterate() changes its position by taking the points from the counter'th frame. Counter is then iterated.

draw() draws its hands and feet, with the position being determined by a centre, and the width and height. The width is constant, while the height is the magnitude of the difference between the

position of the torso and the body part. The centre is the torso PVector + half the PVectors that describes goinf from the torso to the body part (i.e. half (body part - torso))

1.5 Intelligent Objects

The fish and rod are similar. They both have variables for their height, and width. The location and velocity are stored in PVectors. They both have a boolean to track whether they are bouncing (interacting with the jellyfish) called bounce. They both have an arrayList called corners that lists their corners. They also all have PImage variables to store their images. The fish has a pulled boolean while the rod has pulling, which represents them in the act of being pulled up and out of the water.

Their movement is in move(). They both set bounce to false to refresh that state before checking it later in the main file's draw(). For fish : Its x and y velocity stays the same, increases, or decreases by up to 1. It also has a small chance to reverse the x and y direction (independent of each other). Finally, the location has the velocity added to it. This means a negative velocity would reduce the location value. The rod changes its velocity.x to be 3 in the direction of the fish by finding whether the location.x of the fish is in the positive or negative direction of it. The velocity changes, like the fish, but can change by up to 2. It too, then adds the velocity to the location.

Both have wall(). Fish makes sure its center doesn't go past the walls, and when it does, sets it back at the wall, reverses its direction in the axis that concerns it (x for left and right, y for up and down), then halves that axis' velocity. Rod is similar, but since it will find fish as it uses move(), only the top and bottom walls are measured.

fish has face() to check which direction is facing (based on direction of velocity.x)

Rod has makeCorners() which uses its height and width variables, h and w, to generate the position of its corners and stores them in corners.

1.6 Events

The body's hitfish(Fish b) and hitRod(Rod b) checks for whether its corners are in the fish/rod, then if the corners of the fish/rod is within each part of the body. If so, it is true, and it causes the bounce() command on the fish/rod. It does this for each body part apart from the torso by :

- Find vector from torso to body part
- normalizing it then multiplying it by 5, half the width of the body part
- creating an Arraylist of Pvectors to represent the corners in order from one point to an adjacent corner, to that corner's adjacent corner, and so on.
- checking if these corners are within the bounds of the fish/rod b based on their corners ArrayList, and if so, returning true
- checking if each of b's corners being shifted up would cross the edges (represented by one point, and its previous point, though the first point would have its previous point be the last point) only once. This is done by having the float, inter, represent the y value when x = b's corner's x value, then checking to see if that is lower than b's corner's y value. If so, and also if b's corner's x value is within the x value of the edge then it crosses it once.
- if after checking every edge, it only crosses it once, then that b's corner must be in the body part and true is returned.

For the torso, the corners are made by adding or subtracting half the width to x and half the height to y to get the four points. Checking all the corners of torso are within the boundaries of the corners of b, and vice versa lets us tell if that are, and if so, true is returned.

Fish has collide(Rod b) which first makes corners like rod, but then checks if the rod and fish overlap by more than a tiny amount, and if so, changes the pulled and pulling booleans for fish and rod respectively to true. Fish and Rod both have bounce(), which changes the boolean bounce to true, and if their pulled/pulling variable is false, shifts its location to the left for the fish and up for the rod. The fish has its velocity.x set to -5, while the rod has its velocity.y set to -5.

The rod has pull() which represents when when the fish and rod collide, and it ignores fish as it edits both side characters' location.y to move up the image until both are above the top wall, where success is set to true.

Fish and rod have reset() which sets their variables back to the start. Most noticeably their booleans for state are reset.

The main file's draw() determines when the background changes, based on whether a success happened (the rod dragged the fish all the way up).

- If true, it sets background to night, draws then iterates the jellyfish, and decreases the loop variable. When it hits 0, it resets it to 30 (which is 30 frames or 1 second) and resets the jellyfish's counter, and runs reset() on the fish and rod.
- If it isn't a success, the background is set to the daytime underwater imgae, and the program checks for pulling is happening.
  - If it isn't, the fish and rod have move() and wall() run on it, before collide(Rod r) is run by the fish, and makeCorners() is run by the rod to get the corners and tell if they collided so the next frame, if they did, pulling woudl be true. Then, whether hitfish() and hitRod() is true is found, and the corresponding bounce() is run if that is true.
  - If it is pulling, then the rod's pull() is run.

  Now, fish, rod, and jellyfish are drawn, and the jellyfish is iterated. Remember that this is while success is false; The jellyfish was the only thing drawn when success is true.