SSUI Programming Assignment 1 Writeup

released (*Aug 29, 2024*)
due: Sept 12, 2024; 11:59PM ET (Fri) on Canvas

Submission instructions: please zip the project directory with all your files and submit to the Canvas portal.

Task: Implement a Fitt's Law Tester

Preliminaries

Demo: there is a video on Canvas showing off the desired functionality of the completed assignment. It can be found in:

- `Files > Project, Homework, etc. Related > Project1 Resources > ssui-p1-demo.mp4`

Files you should have

ssui-p1

```
- **p1-skel.ts*
- index.html (*use this to launch project after transpiling*)
- tsconfig.json
- among other files
```

If you've set things up correctly, the only thing you should need to touch is going to be the typescript file, which will be titled `p1-skel.ts`

Grading Criteria

```
1. Functional user interface & game for the Fitt's Law Test
2. Commented & well-structured code (rule of thumb: if your code is >= 5 lines and not
immediately self evident, explain what it does)
```

This assignment is pretty simple. Get the code to run and play the game as intended, and you will score pretty well overall. To get the highest marks, we do ask that you explain the logic for your underlying code, and so we'll hold back some amount of points for general good code hygiene.

Read the code for `UIObject` and `ScreenObject`

All of the code to be implemented for this assignment depends on understanding of the the parent classes `UIObject` and `ScreenObject`. It is highly recommended that you read through the code for these classes before attempting to write any code.

Please ask on Slack with any questions about these classes!

Tasks for P1

All of the places where you need to implement code will be marked by a

```
// === YOUR CODE HERE ===
```

`class FittsTestUI`

If you look through the code, you'll have base classes like UIObject and ScreenObject which are already defined for you. Please please do read through the code so you're familiar with these classes before going on to implement subclasses which extend these classes.

`FittsTestUI` is the first thing that you'll run into that you'll be expected to implement.

The basic premise of the UI or game state is that it has 4 total states

- **start**: display a background screen with instructions
- **begin_trial**: display a Reticle which requires the user to place their mouse cursor on -- user should have their mouse pointer directly in a small circle
- **in_trial**: display a random sized Target - this should look different than the Reticle
- **ended**: shows an information screen that the game has ended

(you can find these states in the `switch` statement) you'll be expected to hold, assign, store, modify data to some degree every time there's a state change.

`class Target (extends ScreenObject)`

You'll also be implementing the subclass `Target`, which represents the green blobs that you saw in the demo video.

'newGeom()'

-- where you set the parameters for the bounding box surrounding the object

'draw()'

-- where you write the code to actually how to represent the object on the screen

'pickedBy()'

-- which checks whether your cursor selection is indeed within the Target's dimensions
Remember, the Target is a circle, so it should **not** count as a valid 'pick' if you're in the bounding box, but not in the circle!

'handleClickAt()'

-- where we advance the game state to the next stage

`class Reticle (extends Target)`

Then we have Reticle, which is the specific target type which we click on to start a trial. We have some similar functions to write for the `Reticle` class - (it looks and acts different!)

'draw()'

-- where you write the code to actually how to represent the object on the screen

'pickedBy()'

-- which checks whether your cursor selection is indeed within the Target's dimensions
Remember, the Target is a circle, so it should **not** count as a valid 'pick' if you're in the bounding box, but not in the circle!

'handleClickAt()'

-- where we advance the game state to the next stage
-- remember we're at the **beginning of the trial** so the game state advancing might be different

`class BackgroundDisplay (extends ScreenObject)`

And like before, we have similar functions for you to implement for `BackgroundDisplay`

'draw()'

-- where you display instructions

'handleClickAt()'

-- advance the game state