CS 1410: Frogger

The arcade game Frogger is a classic game from the 1980s. In this game, the player controls a frog, which can hop around the screen. The frog is attempting to return to his home at the top of the screen. The frog must cross lanes of moving vehicles and water in order to safely reach its home. If it is run over by a car, or sinks in the water, it dies.

Assignment

In this assignment, you will build a version of the Frogger game with the most basic features, using the library of classes in `froggerlib`. You will create a class to represent the game, create instances of frogs, cars, logs, etc.

Your program must have at least the following features:

- Must be a *stage* lane where the frog spawns, and another between the traffic and floating objects.
- Frog must be able to cross at least 4 lanes of traffic.
- At least 2 lanes must travel left, and at least 2 lanes must travel right.
- Frog must die and game over if frog touches a vehicle.
- Frog must be able to cross at least 4 lanes of floating objects.
- At least 2 lanes must travel left, and at least 2 lanes must travel right.
- Frog must die and game over if frog is completely surrounded by water.
- Frog must die and game over if frog touches the side walls.
- Must be a lane of at least 3 frog homes.
- Frog must die and game over if frog touches the grass next to home.
- Frog must win and game over if frog completely surrounded by home.
- Not all vehicle or floating lanes can have the same kind of vehicle or floating object.
- A variety of speeds must be used.

Code Requirements

Your software implementation must meet these programming requirements:

- The `froggerlib` code must not be changed.
- The `pygame` framework we've used in other assignments must be used.
- You must have a class to represent the game, and store instances of the objects that make the game.

Milestones

Part1

- Frog controlled by user, moves
- Frog dies if goes off screen
- Both stages drawn
- All traffic lanes drawn
- All traffic lanes populated `Dodgeable` objects
- Vehicles come back on other side of lane when they go off screen
- Frog dies if touches any `Dodgeable` object

Part2

- All water lanes drawn
- All water lanes populated `Rideable` objects
- `Rideable` come back on other side of lane when they go off screen
- Frog dies if completely surrounded by water
- Frog rides `Rideable` objects, if touching them
- All grass and home objects present
- Frog dies if touches grass object
- Frog wins if completely surrounded by home object

Resources

- Download `froggerlib`
- UML of `froggerlib`
- PyGame Starter

Extra Challenges

- Restart option.
- Timer to limit game time.
- Allow frog to complete level by filling all homes.
- Additional levels of increasing difficulty.

Hints

Implement your code in small steps. Make sure each step is working before moving to the next step. For example, this might be a series of steps you could do to get started:

- Get a blank starter kit, maybe by removing code from a previous assignment.
- Unpack `froggerlib` into the folder with your code. `froggerlib` should be folder inside of your folder.
- Create a `frog` object and store it in your game class as a data member.
- Draw the frog every frame.
- Allow the user to cause the frog to move.
- If the frog leaves to window, cause the game to end.
- Build and draw the two stages.
- Build and draw the roads.
- Build one car in one lane.
- Make the car draw and move.
- When the car crosses the screen, move it back to the other side.
- When the frog hits the car, cause the game to end.
- Add more cars to the lane.
- Add more lanes of cars.
- ...

`froggerlib` Notes

- The library uses inheritance and polymorphism abundantly.
- The `hits()` method for `Water` is `True` if the frog is completely surrounded by water, and not riding anything.
- The `hits()` method for `Grass` is `True` if the frog is touched by grass, and not riding anything.
- The `hits()` method for `Home` is `True` if the frog is completely surrounded by home, and not riding anything.
- The `hits()` method for `Dodgeable` objects is `True` if the frog is touched by the object.
- The `hits()` method for all others objects is `False`.
- If the frog `hits()` a `Water`, `Grass`, or `Dodgeable`, the game is over, the player loses.
- If the frog `hits()` a `Home`, the game is over, the player wins.
- Movement is handled the `Movable` class. An object has a current position `x`:`y`, a desired position `desired_x`:`desired_y` and a `speed`. Every time that `move()` is called on a `Movable` object, if the desired position isn't the same as the current position, the object will move up to `speed` pixels towards the desired position.
- Calling and of the `up()`, `down()`, `left()`, `right()` methods on a `PlayerControllable`, like a `Frog`, does not actually move the object. It sets the desired position correctly to cause the object to move towards the desired position when `move()` is called.
- `outOfBounds()` is useful to check if the frog has left the screen. If the frog leaves the screen, the game is over and the player loses.
- Horizontal and vertical gaps define the size of jump the frog will take. They should be set to the size of the lanes.
- The size of objects like `Frog`s, `Car`s, etc, should be smaller than the lanes, so that objects don't `hit()` when they are in neighboring lanes.

Links

- Read About frogger
- Play Frogger
- Wikipedia on Frogger
- Play Frogger