Centipede Sample Design
CSCI 134

This is one way to structure the program, but certainly not the only way. It covers the main properties of the game, but is not a complete implementation --- there may be additional useful methods, instance variables, constants, etc. to include in the code.

Feel free to use ideas from this design in addition to your original designs while working on the code.

---

class Segment

Instance Vars:

- `private static final int SIZE = 16` Width/height of Segment pic
- `private static final int SPEED = 4` horizontal speed for moving

- `VisibleImage body` The segment graphical object

- `boolean goingEast` Is the segment moving east or west?

- `Field shrooms` The mushroom field through which the segment is moving

- `Zapper zapper` The zapper the segment is attacking

Constructor:

```
Segment(Image segmentPic, Location point, Zapper aZapper,
        Field aShrooms, DrawingCanvas aCanvas)
```

Create a segment at the given point with the given picture. Remember the shrooms and zapper for later.

Methods:

- `public void step()`
  if (goingEast) {
    if about to hit mushroom or east edge of field, move down and set goingEast to false;
    else move SPEED pixels to right
  } else {
    if about to hit mushroom or west edge of field, move down and set goingEast to true;
    else move SPEED pixels to left

}
if segment is showing and overlaps the zapper, kill the zapper.

- `boolean contains(Location point)`
  return true if the segment contains the point

- `public void kill()`
  hide the segment image

- `public boolean isAlive()`
  return true if the segment image is showing

- `public double getX()`
- `public double getY()`
  return the x and y coordinates of the segment

---

class Centipede extends ActiveObject

Instance vars:

- `Segment[] segments` Array of Segments making up the centipede
- `Field shrooms` Field of shrooms on the screen
- `Zapper zapper` Zapper we are attacking
- `ScoreKeeper score` The game's score keeper

Constructor:

```
public Centipede(Image segmentImage, int numSegments,
                 Zapper aZapper, Field aShrooms,
                 DrawingCanvas aCanvas)
```

Create the segments array and all the segments so that they are all just off the top left corner of the screen. Initialize the other instance variables with the values passed to the constructor. Call start().

Methods:

- `public void run()`
  while (the game is not over) {
  move each segment
  pause
  if (all segments are dead) tell the scorekeeper the game is over
  }

- `public boolean tryToHitSegment(Location point)`

If point is within an alive segment, kill that segment, make a mushroom appear in its place, and return true. Remember: don't remove the segment from the array --- just tell it to hide its image. Return false if no segments are hit.

---

class Field

Instance vars:

- `private static final int SHROOM_SIZE = 16` Width/height of shroom pics
- `VisibleImage[][] shrooms` 2D array of shroom pics, some of which will be hidden

Constructor:

`public Field(Image shroom, int numShrooms, DrawingCanvas canvas)`
Create the array of mushrooms based on the canvas size and SHROOM_SIZE. Fill it with hidden shroom VisibleImages. Then randomly pick numShrooms of those VisibleImages to show.

Methods:

- `public boolean overlapsShroom(Location point)`
  Returns true if point is contained in a shroom that is shown on the screen. Return false if point is outside of the Field boundaries or within a hidden shroom.

- `public void showShroom(double x, double y)`
- `public void hideShroom(double x, double y)`
  (x,y) must be a valid location in the Field's boundaries. Show/hide the mushroom containing the point (x,y).

- `private int getRow(double y)`
- `private int getColumn(double x)`
  Helper methods provided to convert screen coordinates to column and row indexes.

---

class Missile extends ActiveObject

Instance vars:

- `Line missile` missile on the canvas
- `Field shrooms` Field of shrooms on the screen
- `Centipede bug` Centipede we are shooting at
- `ScoreKeeper score` The game's score keeper

Constructor:

```
public Missile(Location loc, Field aShrooms, ScoreKeeper aScore,
               Centipede aBug, DrawingCanvas canvas)
```

Create the missile line at the given location, and set up instance variables.
Call start().

Methods:

- `public void run()`
  Move the line up the screen until it goes off the top, hits a shroom, or hits the bug.
  Then remove line from screen, and update score appropriately if it hit a shroom or bug.

---

class Zapper

Instance vars:

- `FilledRect body` Objects showing zapper on screen
- `FilledRect gun`

- `ScoreKeeper score` Scorekeeper for game
- `DrawingCanvas canvas` the canvas

Constructor:

```
public Zapper(Location pt, ScoreKeeper aScore,
              DrawingCanvas aCanvas)
```

Create zapper graphical objects and initialize rest of instance vars.

Methods:

- `public void left()`
- `public void right()`
  move to the left/right, but not off the screen

- `public void shoot(Field shrooms, Centipede bug)`
  Create a new missile with the given shroom field and bug. Pass the scorekeeper to the missile's constructor too.
- `public boolean overlaps(VisibleImage image)`
  Return true if the zapper overlaps the given image.

---

class ScoreKeeper

Instance vars:

- `int score` current score
- `Jlabel messageLabel` label showing message on bottom of screen
- `boolean gameOver` true if the game is over because no segments left or zapper killed

Constructor:

`public ScoreKeeper(JLabel aMessageLabel)`
Initialize score to 0 and gameOver to false. Display an initial message in the label.

Methods:

- `public void increment(int amount)`
  If the game is not over, add amount to score and change the message

- `public void endGame()`
  Display game over message and set gameOver to true

- `public boolean isGameOver()`
  Return whether or not the game is over.

---

class CentipedeController extends WindowController implements KeyListener

Instance vars:

- `Field shroomField`
- `Centipede centipede`
- `Zapper zapper`
- `ScoreKeeper score`

Methods:

- `public void begin()`
  Create the black background and the objects stored in the instance variables. Also create the JLabel for the south part of the screen that will be used by the score keeper.

- `public void keyPressed(KeyEvent event)`
  Tell the zapper to move/shoot in response to the various keystrokes.
  If the score keeper says that game is over, do nothing.