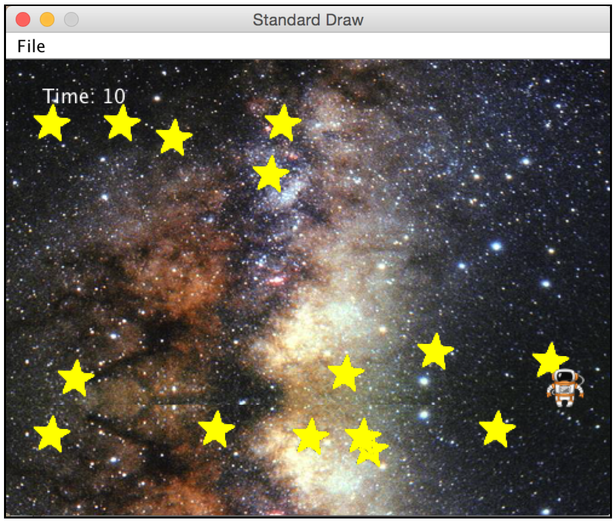
University *of* New Orleans - Department *of* Computer Science

Star Dodger

A High Speed Dodge-style Arcade Game



Homework Summary

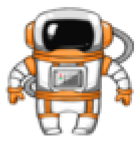
iterations	**Overview**	Expected Time	Points per Goal (Sum)
Goal 0	**Planning:** *What Game Objects are needed?*	10 minutes	+5 points (05)
Goal 1	**Game class:** *Architecture of Main Game Logic*	10 minutes	+5 points (10)
Goal 2	**Scene class:** *Draw the graphics for the Scene*	10 minutes	+5 points (15)
Goal 3	**Enemy class:** *Create & Draw Many Asteroids*	10 minutes	+5 points (20)
Goal 4	**Enemy class:** *Move All Asteroids*	10 minutes	+5 points (25)
Goal 5	**Game class:** *Duration between Enemy Spawns*	10 minutes	+5 points (30)
Goal 6	**Player class:** *Draw the player*	10 minutes	+5 points (35)
Goal 7	**Controller class:** *Player Controls, Move Player*	10 minutes	+5 points (40)
Goal 8	**Player class:** *Collision Detection*	10 minutes	+5 points (45)
Goal 9	**Game class:** *Heads-Up Display, Game Timer*	10 minutes	+5 points (50)
	Homework Assignment	∞ minutes	+50 points (100)

By Ted Holmberg *(eholmber@uno.edu)*

Introduction:
In this homework, you'll build a space dodger game. The player must avoid the falling stars as they travel through space. The game must generate hundreds of enemies. To implement such a game, you must use an Object Oriented Programming approach, where the classes contain constructors that create separate, autonomous instances. Each instance can represent a single enemy. This way just one class can be used to build any number of new enemy objects.

Quick Explanation of Software Objects:
Classes may contain non-static data and methods: *instance variables* and *instance methods*. Instance variables are initialized by invoking a class's constructor method with the *new* keyword. Each class instance may be stored in either variable or an array.

Game Objects (from player perspective):

	Spaceman: Player character
	Asteroid: Enemy character, Player dies if they collide into a Star.

Version 0: Goal: Designing the Game classes.

`Class`	Description	methods
`Game`	Manages Game data and game methods	main, start, update, render
`Scene`	Manages Scene data and Scene methods	start, draw
`Player`	Manages Player data and Player methods	start, draw
`Enemy`	Manages Enemy data and Enemy methods	start, draw
`Controller`	Manages Player inputs (Static class)	
`StdDraw`	Library Class for drawing graphics	

GitLab Repo: (Starter files for this Homework)
https://gitlab.com/scalemailted/dodgergame-files

Goal 1: Create a Game object.

DodgerGame.java → Class & instance variables

```
public class DodgerGame {
    private boolean isOver;
}
```

DodgerGame.java → Constructor (instance method)

```
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
}
```

DodgerGame.java → update (instance method)

```
public void update() {
    //game update code
}
```

DodgerGame.java → render (instance method)

```
public void render() {
    //game draw code
}
```

DodgerGame.java → main (class method)

```
/* The main game loop*/
public static void main(String[] args) {
    DodgerGame game = new DodgerGame();
    while (game.isOver == false) {
        game.update();
        game.render();
    }
}
```

Output:
Compile code, make sure no errors or exceptions occur.

Goal 2: Game object creates & draws a new Scene object

Scene.java → Class and instance variables

```
public class Scene {
    /*Attributes: Instance Variables*/
    private String image;
    private int width;
    private int height;
}
```

Scene.java → constructor (instance method)

```
/* Constructor */
public Scene() {
    width = 500;                                        //set scene width
    height = 350;                                       //set scene height
    image = "assets/space-background.png";              //set image filename

    StdDraw.setCanvasSize(width, height);                    //set canvas size for image
    StdDraw.setXscale(0.0, width);                           //set x=0 from right to left
    StdDraw.setYscale(height, 0.0);                          //set y=0 from top to bottom
}
```

Scene.java → draw (instance method)

```
/*draw method*/
public void draw() {
    StdDraw.picture(width/2, height/2, image);   //draw background image using center point
}
```

DodgerGame.java → instance variables

```
/*Dodger Game Attributes*/
private Scene scene;
private boolean isOver;
```

DodgerGame.java → constructor

```
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
    scene = new Scene();
}
```

DodgerGame.java → render()

```
public void render() {
    scene.draw();                          //draw scene
    StdDraw.show(100);
}
```

Output:
Compile and run, you should see the background image display.

Goal 3: Scene creates multiple Enemy objects at random coords & draws them.

Enemy.java → instance variables

```java
public class Enemy {
    /*Attributes: Instance Variables*/
    private String image;
    private int width;
    private int height;
    private double x;
    private double y;
}
```

Enemy.java → constructor

```java
/* Constructor */
public Enemy(double x, double y) {
    this.x = x;                                                 //set x position
    this.y = y;                                                 //set y position
    this.width = 32;                                            //set width
    this.height = 32;                                           //set height
    this.image = "assets/asteroid.png";                         //set image filename
}
```

Enemy.java → draw()

```java
public void draw() {
    StdDraw.picture(x, y, image, width, height);
}
```

Scene.java

```java
import java.util.ArrayList;
```

Scene.java → instance variables

```java
/*Attributes: Instance Variables*/
private String image;
private int width;
private int height;
private ArrayList<Enemy> monsters;
```

Scene.java → constructor

```java
/* Constructor */
public Scene() {
    width = 500;                                    //set scene width
    height = 350;                                   //set scene height
    image = "assets/space-background.png";          //set image filename
    monsters = new ArrayList<Enemy>();              //set a new list of enemies in this scene

    StdDraw.setCanvasSize(width, height);                //set canvas size for image
    StdDraw.setXscale(0.0, width);                       //set x=0 from right to left
    StdDraw.setYscale(height, 0.0);                      //set y=0 from top to bottom
}
```

Scene.java → draw()

```
public void draw() {
    StdDraw.picture(width/2, height/2, image);
    for (Enemy monster : monsters) {
       monster.draw();
    }
}
```

Scene.java → addMonster()

```
public void addMonster() {
    double x = 32 + (Math.random() * (width - 64));
    double y = Math.random() * height;
    Enemy star = new Enemy(x,y);
    monsters.add(star);
}
```

DodgerGame.java → update()

```
public void update() {
    scene.addMonster();
}
```

Output:
Compile and run, asteroids should appear at random locations in the scene.

Goal 4: Move all the Asteroid objects within the Scene object

Enemy.java → instance variables

```
/*Attributes: Instance Variables*/
private String image;
private int width;
private int height;
private double x;
private double y;
private int speed;
```

Enemy.java → constructor

```
/* Constructor */
public Enemy(double x, double y) {
    this.x = x;                                         //set x position
    this.y = y;                                         //set y position
    this.width = 32;                                    //set width
    this.height = 32;                                   //set height
    this.image = "assets/asteroid.png";                 //set image filename
    this.speed = (int) (3 + Math.random() * 10);
}
```

Enemy.java → move()

```
public void move() {
    this.y += this.speed;
}
```

Scene.java → addMonster()

```
public void addMonster() {
    double x = 32 + (Math.random() * (width - 64));
    double y = -32;
    Enemy star = new Enemy(x,y);
    monsters.add(star);
}
```

Scene.java → update()

```
public void update() {
   for (Enemy monster : monsters) {
       monster.move();
   }
}
```

DodgerGame.java → update()

```
public void update() {
    scene.addMonster();
    scene.update();
}
```

Output:
Compile and run, the asteroids should spawn at the top of scene and fall to the bottom of scene.

Goal 5: Generate new Asteroid objects after waiting for a 1/5th second duration.

DodgerGame.java → instance variables

```java
private Scene scene;
private boolean isOver;
private long startTime;
```

DodgerGame.java → constructor

```java
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
    scene = new Scene();
    startTime = System.currentTimeMillis();

}
```

DodgerGame.java → update()

```java
public void update() {
    long now = System.currentTimeMillis();
    if (now - this.startTime > 200 ) {             // 1/5th sec duration //
        scene.addMonster();
        this.startTime = now;
    }
    scene.update();
}
```

Output:
Compile and run, the asteroids should spawn at a slower rate (5 asteroids / 1 sec).

Goal 6: Create and draw a Player object into the Scene object
Note: use composition to contain a player object within a scene object

Player.java → instance variables

```
public class Player {
    /*Attributes: Instance Variables*/
    private String image;
    private int width;
    private int height;
    private double x;
    private double y;
}
```

Player.java → constructor

```
/* Constructor */
public Player(double x, double y) {
    this.x = x;                                         //set x position
    this.y = y;                                         //set y position
    this.width = 32;                                    //set width
    this.height = 32;                                   //set height
    this.image = "assets/spaceman.png";                 //set image filename
}
```

Player.java → draw()

```
public void draw()  {
    StdDraw.picture(x, y, image, width, height);
}
```

Scene.java → instance variables

```
/*Attributes: Instance Variables*/
private String image;
private int width;
private int height;
private ArrayList<Enemy> monsters;
private Player player;
```

Scene.java → getPlayer()

```
public Player getPlayer() {
    return this.player;
}
```

Scene.java → setPlayer()

```
public void setPlayer( Player player) {
    this.player = player;
}
```

Scene.java → draw()

```
public void draw() {
    StdDraw.picture(width/2, height/2, image);  //draw background image using center point
    for (Enemy monster : monsters) {
       monster.draw();
    }
    player.draw();                              //draw player
}
```

DodgerGame.java → constructor

```
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
    scene = new Scene();
    Player player = new Player(250, 187.5);
    this.scene.setPlayer( player );
}
```

Output:
Compile and run, the Player object should render into the Game Scene

Goal 7: Create Controller object that moves a Player object via mouse clicks

Player.java → instance variables

```java
/*Attributes: Instance Variables*/
private String image;
private int width;
private int height;
private int x;
private int y;
private int speed;
```

Player.java → constructor

```java
/* Constructor */
public Player(int x, int y) {
    this.x = x;                                        //set x position
    this.y = y;                                        //set y position
    this.width = 32;                                   //set width
    this.height = 32;                                  //set height
    this.image = "assets/spaceman.png";                //set image filename
    this.speed = 10;                                   //set speed
}
```

Player.java → move()

```java
public void move(double x, double y) {
    this.y = y;
    this.x = x;
}
```

Player.java → getters

```java
public double getX() {
    return this.x;
}

public double getY() {
    return this.y;
}

public double getSpeed() {
    return this.speed;
}
```

Controller.java

```java
public class Controller {
    private Player player;
}
```

Controller.java → constructor

```java
public Controller( Player player) {
    this.player = player;
}
```

Controller.java → onMousePress()

```java
public void onMousePress() {
    if ( StdDraw.mousePressed() ) {
        double mouseX = StdDraw.mouseX();
        double mouseY = StdDraw.mouseY();
        double playerX = player.getX();
        double playerY = player.getY();

        if (mouseY < playerY ) playerY -= player.getSpeed();
        if (mouseY > playerY ) playerY += player.getSpeed();
        if (mouseX < playerX ) playerX -= player.getSpeed();
        if (mouseX > playerX ) playerX += player.getSpeed();
        player.move( playerX, playerY );
    }
}
```

Controller.java → update()

```java
public void update() {
    onMousePress();
}
```

DodgerGame.java → instance variables

```java
private Scene scene;
private boolean isOver;
private long startTime;
private Controller controller;
```

DodgerGame.java → constructor

```java
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
    scene = new Scene();
    startTime = System.currentTimeMillis();
    Player player = new Player(250, 187.5);
    scene.setPlayer( player );
    controller = new Controller( player );
}
```

DodgerGame.java → update()

```java
public void update() {
    controller.update();
    long now = System.currentTimeMillis();
    if (now - this.startTime > 200 ) {            // 1/5th sec duration //
        scene.addMonster();
        this.startTime = now;
    }
    scene.update();
}
```

Output:
Compile and run, the Player should move towards the location of mouse clicks

Goal 8: Collision detection and Game Over condition

Enemy.java → getters

```java
public double getX() {
    return this.x;
}

public double getY() {
    return this.y;
}
```

Player.java → isTouchingX()

```java
public boolean isTouchingX( Enemy gameObject ) {
    int hitzone = 24;
    return this.x <= gameObject.getX()+hitzone && gameObject.getX() <= this.x+hitzone;
}
```

Player.java → isTouchingY()

```java
public boolean isTouchingY( Enemy gameObject ) {
    int hitzone = 24;
    return this.y <= gameObject.getY()+hitzone && gameObject.getY() <= this.y+hitzone;
}
```

Player.java → isTouching()

```java
public boolean isTouching( Enemy gameObject ) {
    return  this.isTouchingX(gameObject) && this.isTouchingY(gameObject);
}
```

Scene.java → getters

```java
public ArrayList<Enemy> getMonsters() {
    return this.monsters;
}
```

DodgerGame.java → update()

```java
public void update() {
    controller.update();
    long now = System.currentTimeMillis();
    if (now - this.startTime > 200 ) {                          //1/5th sec duration //
        scene.addMonster();
        this.startTime = now;
    }
    scene.update();
    Player player = scene.getPlayer();
    for (Enemy monster : scene.getMonsters() ) {
        if ( player.isTouching(monster) ) {
            isOver = true;
        }
    }
}
```

Output: Compile and run, Game should end when the Player collides with an Asteroid.

Goal 9: Add timer in HUD

DodgerGame.java → instance variables

```
private Scene scene;
private boolean isOver;
private long startTime;
private Controller controller;
private int timer;
```

DodgerGame.java

```
/*Create a new Dodger Game*/
public DodgerGame() {
    isOver = false;
    scene = new Scene();
    startTime = System.currentTimeMillis();
    Player player = new Player(250, 187.5);
    scene.setPlayer( player );
    controller = new Controller( player );
    this.timer = 0;
}
```

DodgerGame.java → render()

```
public void render() {
        scene.draw();
        StdDraw.setPenColor(StdDraw.WHITE);
        StdDraw.text(64,32,"Time:  " + timer/5 );
        StdDraw.show(100);
}
```

DodgerGame.java → update()

```
public void update()  {
    controller.update();
    long now = System.currentTimeMillis();
    if (now - this.startTime > 200 ) { //1/5th sec duration //
        scene.addMonster();
        this.timer++;
        this.startTime = now;
    }
    scene.update();
    Player player = scene.getPlayer();
    for (Enemy monster : scene.getMonsters() ) {
        if ( player.isTouching(monster) ) {
            isOver = true;
        }
    }
}
```

Homework 6: Dodger Game Enhancements

Part 1 (45 points):

Add three features to the game, any additional features count towards bonus points. Features must be implemented using an Object-Oriented Approach.

1. Collect gems that spawn into the level. Score shows how many you collected

2. Timer counts down as an oxygen counter, if it reaches zero you die. Must collect oxygen tanks to stay alive.

3. Add different types of enemies into the game that damages the player

4. Player can shoot at the asteroids
   +1 powerups for weapon, armor

5. Add multiple levels to the game

6. Improve player movement with mouse using better math
   Move the hero around with the mouse click (shortest path using better math)
   Hint: find the slope between current position and destination position.
   If the slope distance is greater than the player speed, then use speed / distance to find the ratio and multiply by the destination x and destination y coordinate

7. Scroll background scene image to animate it

8. Add 2 player support, mouse controls for player 1 and keyboard controls for player 2

9. End level boss battle after surviving a certain duration

Part 2 (5 Points):

Write a readme.txt file detailing the custom features you added into your version of the game, explain how it works, and how to play.

Submitting:

Submit all homework files to both the lab's gitlab repo and Moodle (zipped file):