Criterion C: Development

Techniques used to create Asteroids.

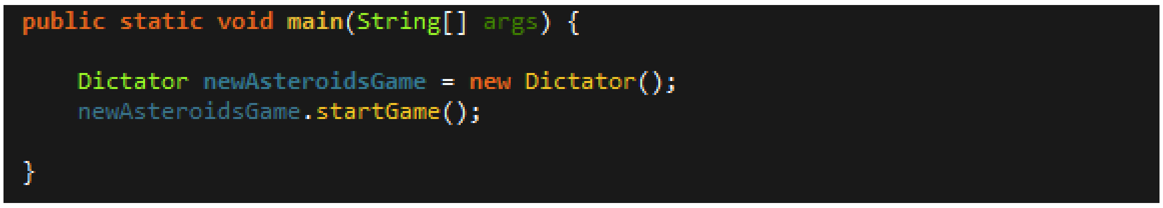
- Game Loop – Page 1, 4
- Graphics and Displaying – Page 6-7

Rest of code is featured in Appendix Classes

Game Loop

For many smaller games, it is easy to use a method called a game loop. This loop usually is a while loop that runs continuously while the application is running. The reason for this type of iterative approach for managing the application is because it allows easy separation between rendering the game, and computation on the game.

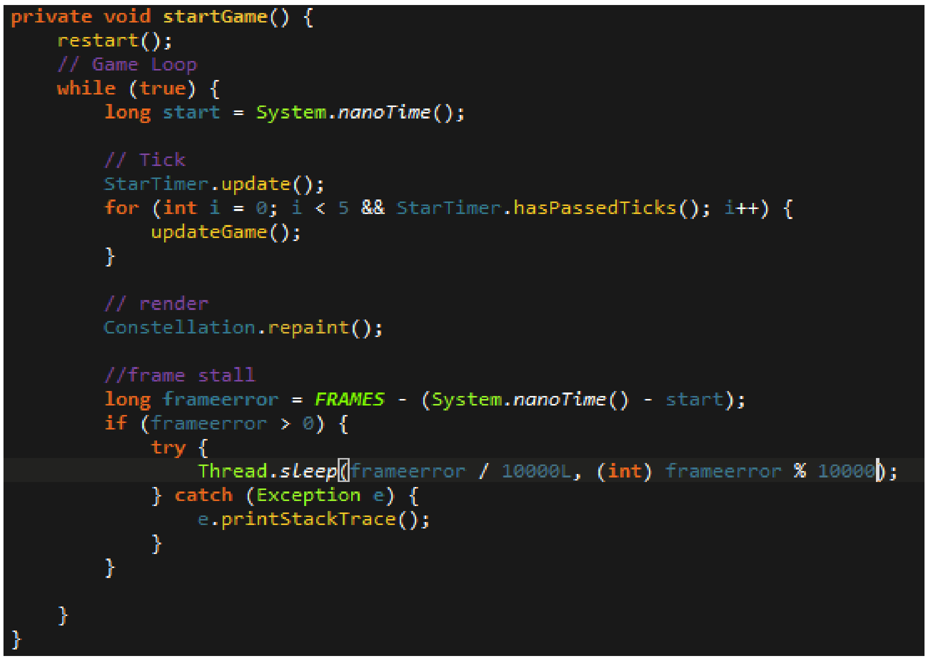
The Game Loop starts when the main calls startGame() on a dictator object. The main method is the method run when the application jar is called, however, since it is static, and all the benefits of java come from object oriented computation, not much happens in the main.



```
public static void main(String[] args) {

    Dictator newAsteroidsGame = new Dictator();
    newAsteroidsGame.startGame();

}
```

Illustration 1: Main Method

As shown in the main, the .startGame() method is called onto a new Dictator object. The startgame method serves to both initiate the game, and start the Game Loop

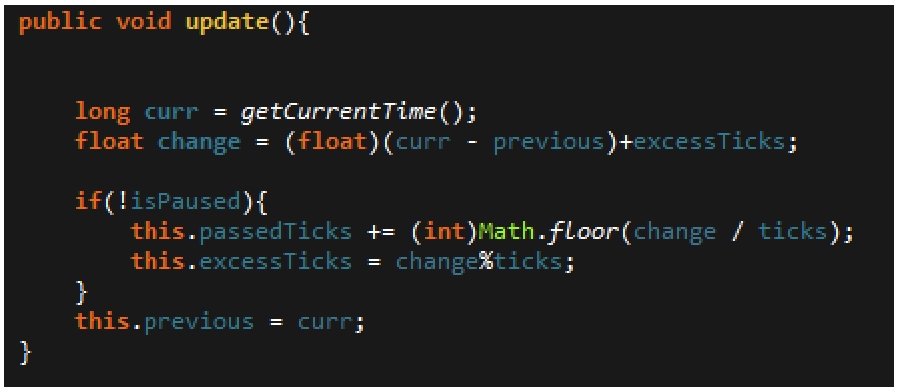


```java
private void startGame() {
    restart();
    // Game Loop
    while (true) {
        long start = System.nanoTime();

        // Tick
        StarTimer.update();
        for (int i = 0; i < 5 && StarTimer.hasPassedTicks(); i++) {
            updateGame();
        }

        // render
        Constellation.repaint();

        //frame stall
        long frameerror = FRAMES - (System.nanoTime() - start);
        if (frameerror > 0) {
            try {
                Thread.sleep(frameerror / 10000L, (int) frameerror % 10000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

    }
}
```

.

Illustration 2: Game Loop Method

The restart() method serves to instantiate the rest of the variables the Dictator class must have e.g. all the menu booleans. This method is kept not part of the constructor because when a restart is called onto the game, instead of remaking a dictator object again, it just runs restart().

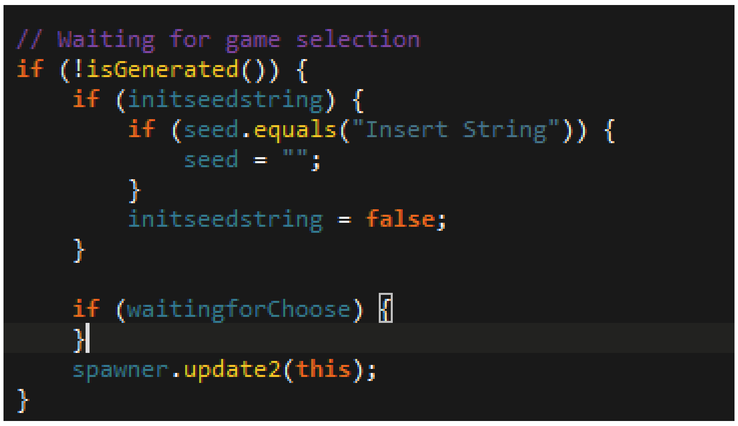


```java
public void update(){


    long curr = getCurrentTime();
    float change = (float)(curr - previous)+excessTicks;

    if(!isPaused){
        this.passedTicks += (int)Math.floor(change / ticks);
        this.excessTicks = change%ticks;
    }
    this.previous = curr;
}
```

Illustration 3: Update Method in Watch method

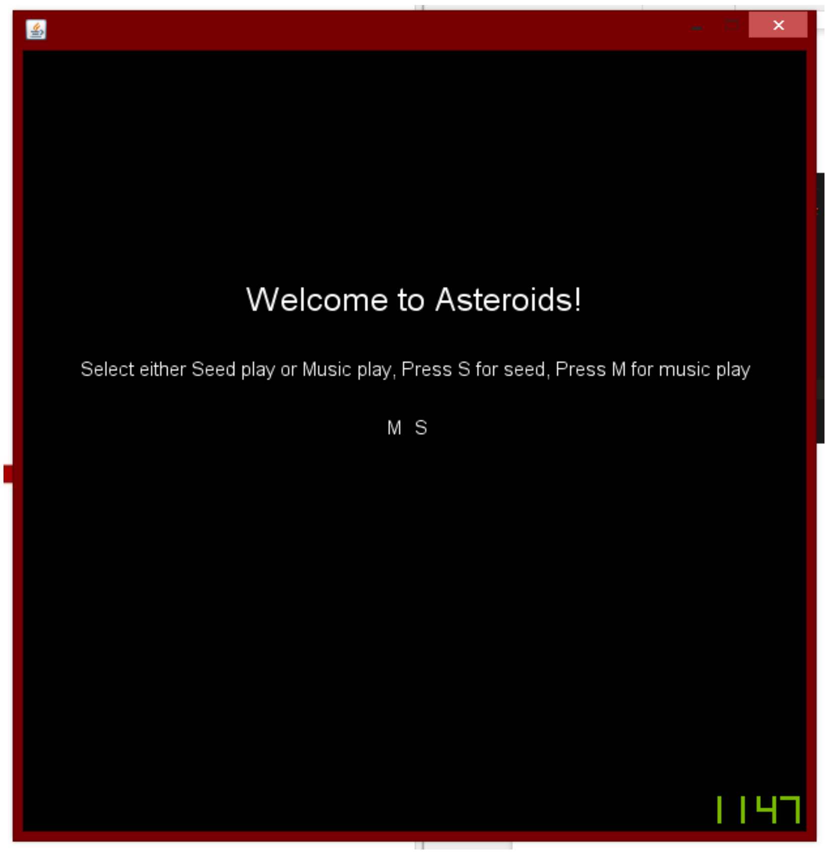
Next we see the game loop start, shown by the while(true) loop. The for loop stalls the game slightly so that the updateGame() or the “tick” method isn't called to fast to often. The Startimer object is a watch object. When update is called, the time on the StarTimer .is updated

After the tick check, comes the updateGame() method. The update method has two main parts to it's functionality. The pre game, and during game segments



```java
// Waiting for game selection
if (!isGenerated()) {
    if (initseedstring) {
        if (seed.equals("Insert String")) {
            seed = "";
        }
        initseedstring = false;
    }

    if (waitingforChoose) {
    }
    spawner.update2(this);
}
```

Illustration 4: Not in game loop, in update()



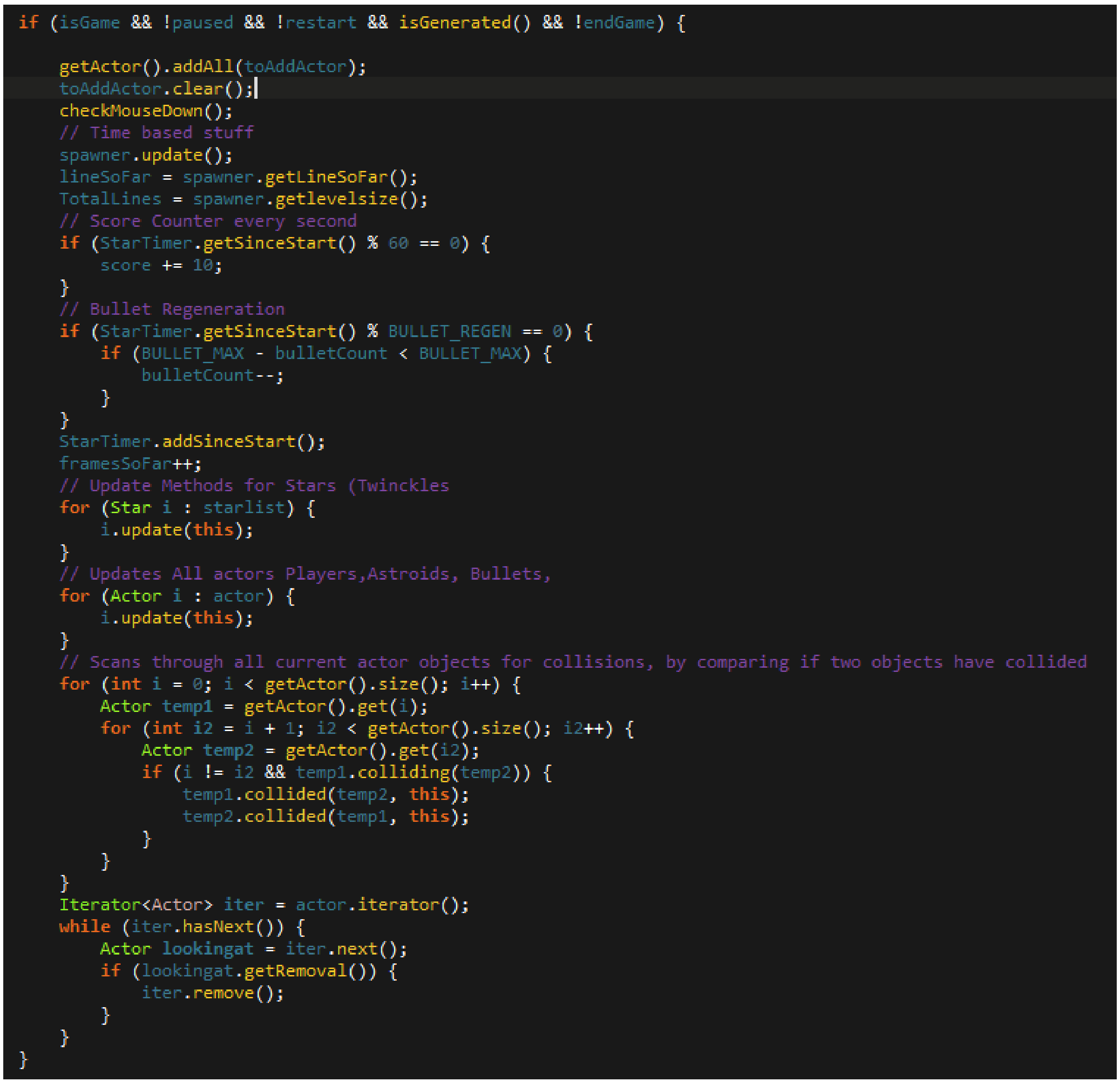
Welcome to Asteroids!

Select either Seed play or Music play, Press S for seed, Press M for music play

M S

Illustration 5: Not in game mode example

The next part is the during game-play loop.



```
if (isGame && !paused && !restart && isGenerated() && !endGame) {

    getActor().addAll(toAddActor);
    toAddActor.clear();
    checkMouseDown();
    // Time based stuff
    spawner.update();
    lineSoFar = spawner.getLineSoFar();
    TotalLines = spawner.getlevelsize();
    // Score Counter every second
    if (StarTimer.getSinceStart() % 60 == 0) {
        score += 10;
    }
    // Bullet Regeneration
    if (StarTimer.getSinceStart() % BULLET_REGEN == 0) {
        if (BULLET_MAX - bulletCount < BULLET_MAX) {
            bulletCount--;
        }
    }
    StarTimer.addSinceStart();
    framesSoFar++;
    // Update Methods for Stars (Twinckles
    for (Star i : starlist) {
        i.update(this);
    }
    // Updates All actors Players,Astroids, Bullets,
    for (Actor i : actor) {
        i.update(this);
    }
    // Scans through all current actor objects for collisions, by comparing if two objects have collided
    for (int i = 0; i < getActor().size(); i++) {
        Actor temp1 = getActor().get(i);
        for (int i2 = i + 1; i2 < getActor().size(); i2++) {
            Actor temp2 = getActor().get(i2);
            if (i != i2 && temp1.colliding(temp2)) {
                temp1.collided(temp2, this);
                temp2.collided(temp1, this);
            }
        }
    }
    Iterator<Actor> iter = actor.iterator();
    while (iter.hasNext()) {
        Actor lookingat = iter.next();
        if (lookingat.getRemoval()) {
            iter.remove();
        }
    }
}
```

Illustration 6: GameLoop during gamemode

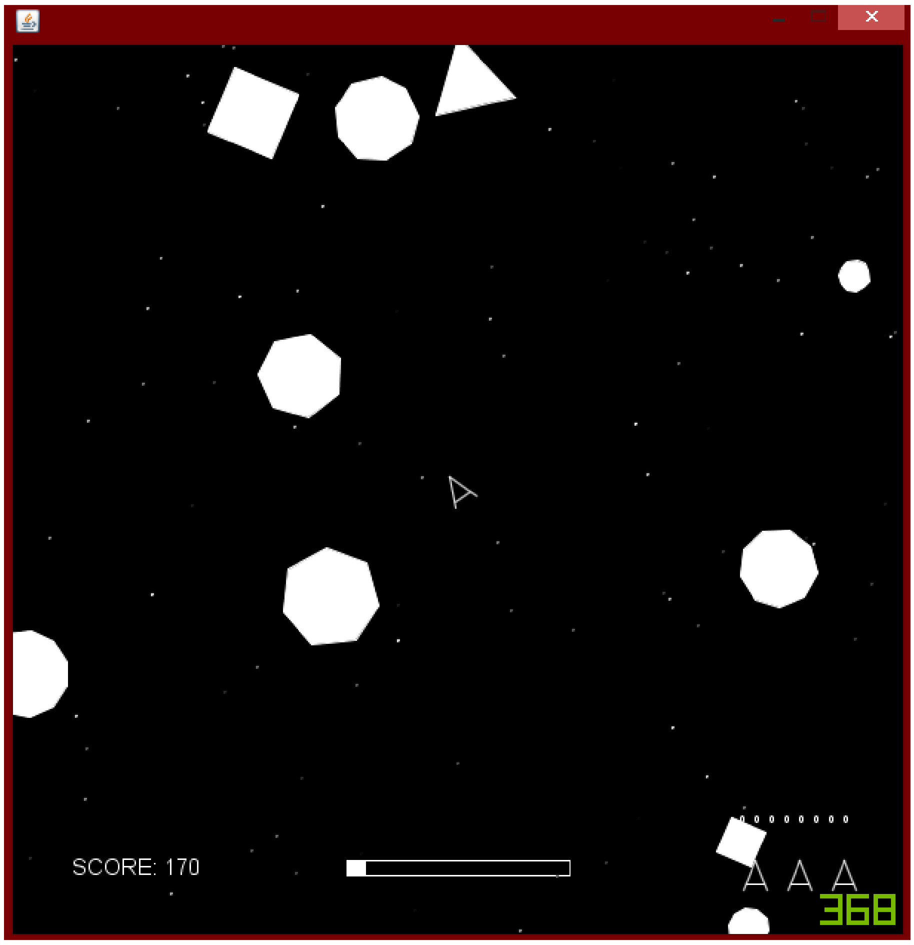
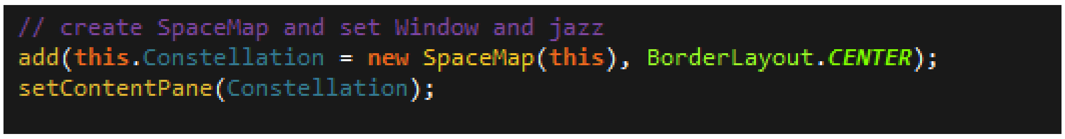


Illustration 7: Game Play Example

Graphics

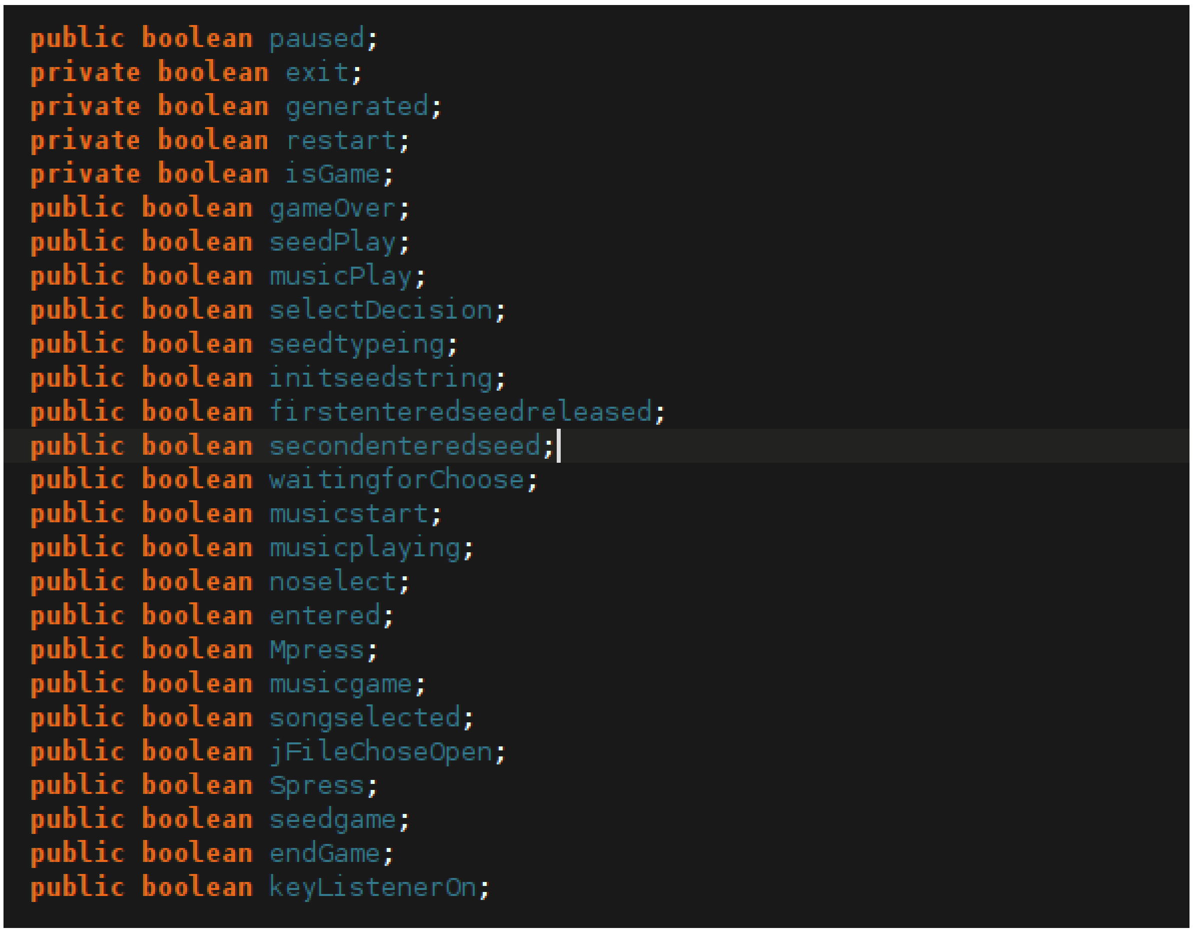
For graphics, I made a separate class called SpaceMap, which is created in the constructor of the Dictator class.



```
// create SpaceMap and set Window and jazz
add(this.Constellation = new SpaceMap(this), BorderLayout.CENTER);
setContentPane(Constellation);
```

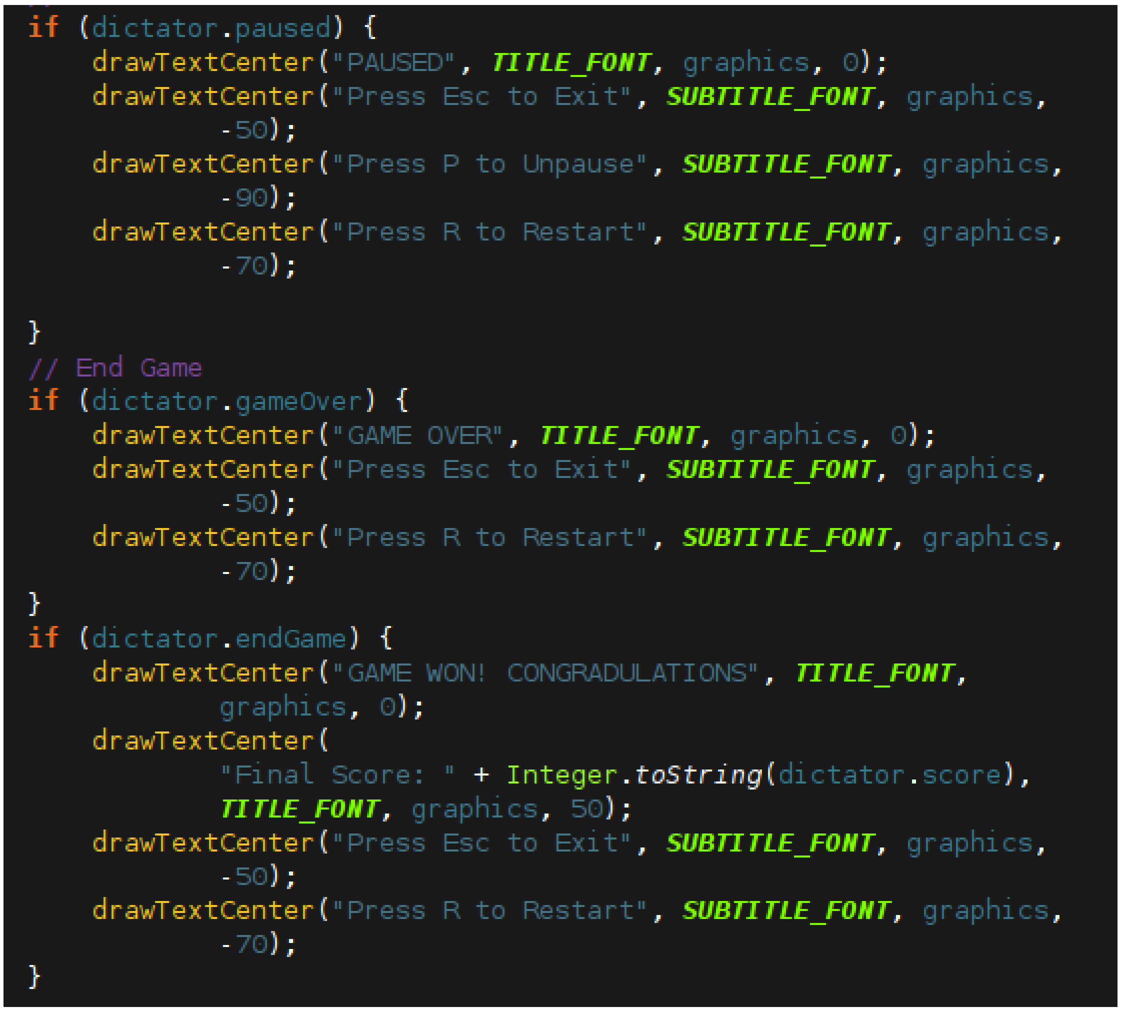
Illustration 8: Makes a Constellation/spacemap object, then sets it to the contentPane

The space map extends Jpanel, another default java class that deals with graphics. Space map, similar to the updateGame() method, has different boolean checkers to decide what to display on the screen. This way, there's separation between rendering a menu, all the game entities, or the pause menu.



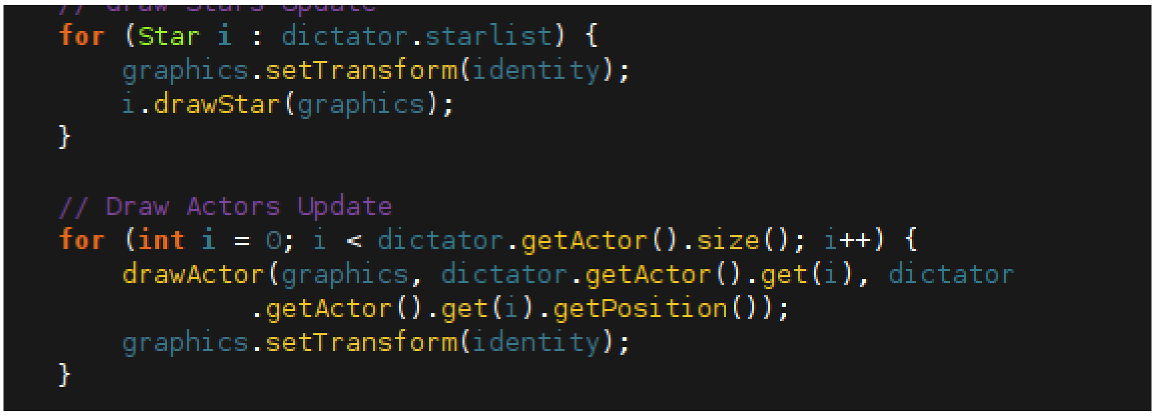
```
public boolean paused;
private boolean exit;
private boolean generated;
private boolean restart;
private boolean isGame;
public boolean gameOver;
public boolean seedPlay;
public boolean musicPlay;
public boolean selectDecision;
public boolean seedtypeing;
public boolean initseedstring;
public boolean firstenteredseedreleased;
public boolean secondenteredseed;
public boolean waitingforChoose;
public boolean musicstart;
public boolean musicplaying;
public boolean noselect;
public boolean entered;
public boolean Mpress;
public boolean musicgame;
public boolean songselected;
public boolean jFileChoseOpen;
public boolean Spress;
public boolean seedgame;
public boolean endGame;
public boolean keyListenerOn;
```

Illustration 9: Booleans for Menu Graphics



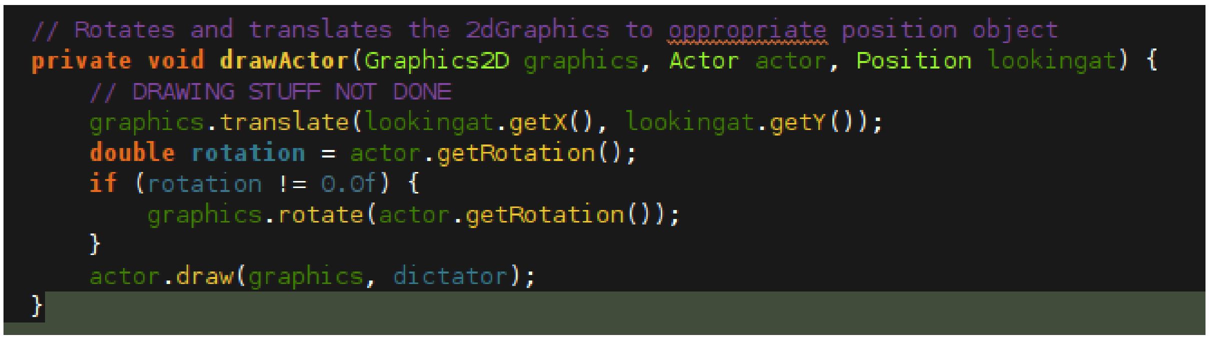
```java
if (dictator.paused) {
    drawTextCenter("PAUSED", TITLE_FONT, graphics, 0);
    drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
            -50);
    drawTextCenter("Press P to Unpause", SUBTITLE_FONT, graphics,
            -90);
    drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
            -70);

}
// End Game
if (dictator.gameOver) {
    drawTextCenter("GAME OVER", TITLE_FONT, graphics, 0);
    drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
            -50);
    drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
            -70);
}
if (dictator.endGame) {
    drawTextCenter("GAME WON! CONGRADULATIONS", TITLE_FONT,
            graphics, 0);
    drawTextCenter(
            "Final Score: " + Integer.toString(dictator.score),
            TITLE_FONT, graphics, 50);
    drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
            -50);
    drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
            -70);
}
```

Illustration 10: If statement checkers for Graphics



```java
for (Star i : dictator.starlist) {
    graphics.setTransform(identity);
    i.drawStar(graphics);
}

// Draw Actors Update
for (int i = 0; i < dictator.getActor().size(); i++) {
    drawActor(graphics, dictator.getActor().get(i), dictator
            .getActor().get(i).getPosition());
    graphics.setTransform(identity);
}
```

Illustration 11: Rendering in Space Map class



```
// Rotates and translates the 2dGraphics to oppropriate position object
private void drawActor(Graphics2D graphics, Actor actor, Position lookingat) {
    // DRAWING STUFF NOT DONE
    graphics.translate(lookingat.getX(), lookingat.getY());
    double rotation = actor.getRotation();
    if (rotation != 0.0f) {
        graphics.rotate(actor.getRotation());
    }
    actor.draw(graphics, dictator);
}
```

Illustration 12: drawActor class