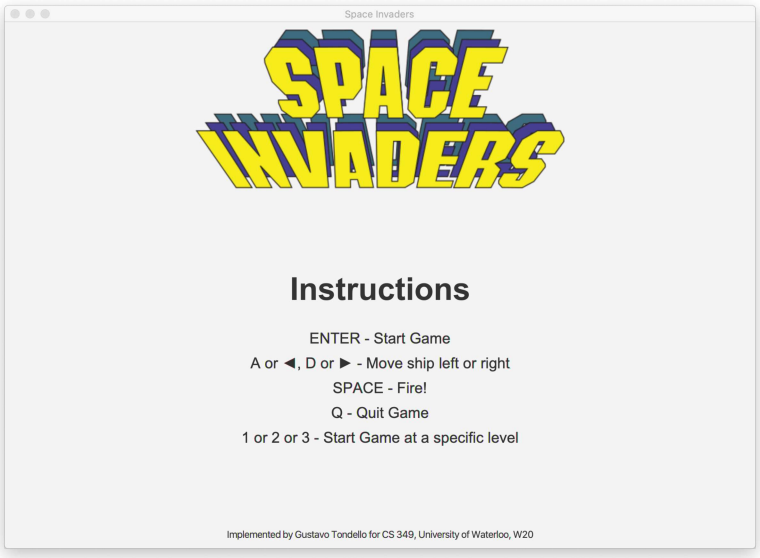
A2: Event Loops and Drawing (Java) -- Space Invaders

Synopsis

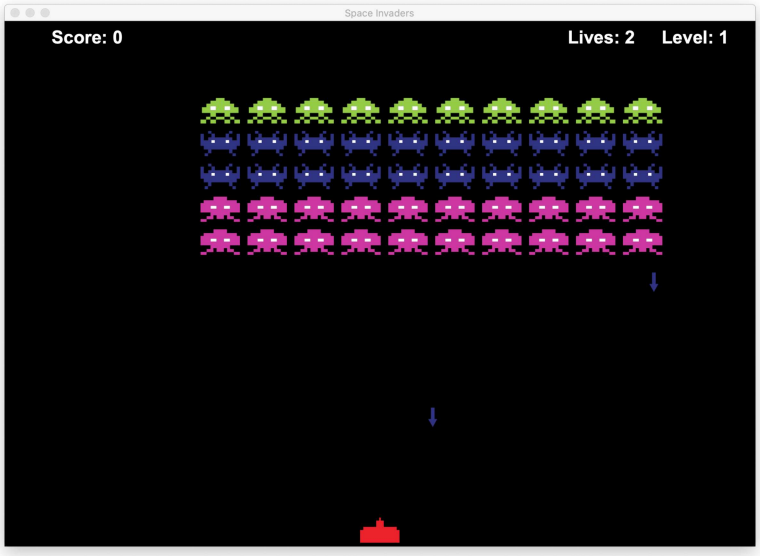
Space Invaders is a simple 2D video game, released by Taito in Japan in 1978 and the US in 1980. It was *hugely popular*, and helped to launch the commercial video game industry. The object of the game is to move a ship along the bottom of the screen and shoot up at aliens that are descending from the sky, without being hit in return. For this assignment, you will implement a fun, playable, version of Space Invaders in Java/Java FX.

For motivation, here's a YouTube video of the original US Atari release (https://youtu.be/MU4psw3ccUI) of the game, and a screenshot of the original game screen. (space-invaders-original.png)

We've also made our own version that matches the assignment specification. Here's a video for reference (space-invaders-demo.mp4), and screen shots are below. Your game should closely resemble this!



Title screen



Game screen

Objectives

a. Learn how to build an interactive application in Java.

b. Use JavaFX to handle drawing and animation of simple graphics.

c. Provide visual and audial feedback, and gameplay elements to make the game more enjoyable.

Features

Starting the Game

When launched, you should display a title screen including (at a minimum):

a. The title, "Space Invaders", at the top of the screen,

b. Your name and student number, somewhere on screen,

c. Instructions on how to play and/or quit your application.

The player should be able to start or quit the game from this screen.

When the game starts, it should resemble the screenshot above, with standard gameplay elements:

a. a game screen, including at least 10 columns and 5 rows of aliens,

b. a player-controlled ship that can move along the bottom of the screen,

c. a high-score indicator,

d. a level indicator to show which level is in-play, and

e. an indicator of the number of ships remaining.

Status information (high score, level, number of ships) should be visible during gameplay, without obscuring the board.

The Aliens

The alien ships move together as a group. They initially move from their starting position towards the right-side of the screen. When they reach the edge of the screen, the following happens:

1. The ships all descend one row,
2. One of the ships fires a missile straight down,
3. The ships reverse their direction and start moving in the opposite direction.

The ships continue moving until they reach the left-edge of the screen, and the pattern repeats and they start moving right again. They repeat this pattern, descending and alternating directions, until they reach the bottom of the screen or the game ends.

Every time the aliens move, there is a random chance that one of them fires a missle. You should design the rate-of-fire so that there are never more than a few missiles on-screen at a given time.

Additionally, every time the player destroys an alien, the remaining aliens speed up. Effectively the game starts out fairly slow, and gets progressively faster as aliens are destroyed.

Game Progression

The player has three ships at the start of the game: the ship they are using and two additional ships which are indicated on the screen. If the player has ships remaining and their ship is destroyed (either by contacting an alien or being struck by a missile), you should remove one of the extra ships and respawn the player's ship in a random, unoccupied location on-screen. If they die for a third time (i.e. they've exhausted all ships), then they lose the game.

If all of the aliens are destroyed successfully, the next level is launched. Subsequent levels all have the same layout, but the ships and missiles move faster than previous levels, and there may be a slightly higher chance of missiles firing in later levels. The level indicator should clearly indicate the level in play. There does not need to be any other visual changes.

You must have a minimum of three levels, and pressing "1", "2" or "3" from the start screen will launch the game at that level (this is not normal gameplay, but makes it easier for the TA to grade it!).

If the player succeeds in clearing all three levels, you should show a message telling them that they have won the game and display their high score. You should prompt them to restart or quit the game.

If the player's ships are all destroyed, you should display a message telling them that they have lost the game, along with their high score. You should prompt them to restart or quit the game.

Controls

The player interacts with the game by moving the ship left and right to avoid alien missiles, while attempting to return fire: 'A' moves the ship left, 'D' moves the ship right, and SPACE fires missiles. The player may fire missiles as often as they wish (with the rate of fire not exceeding 2 missiles per second, and it's possible for multiple missiles to be in motion). Missiles fly straight up until they strike an alien ship. Missiles will destroy any alien that they strike. Missiles that do not hit an alien disappear when they reach the top of the screen. The player effectively has unlimited ammunition.

Game Features

Your game screen should be 800x600 pixels. Your game window should NOT be resizable.

Your game should use audio feedback/sounds for key game elements (e.g. movement of the aliens, when missiles are fired, when an alien ship is destroyed).

You should use images for all of the game elements.

You have a number of choices for sound and image assets:

- This Space Invaders fan site has the original image files and sound clips (http://www.classicgaming.cc/classics/space-invaders/). You may use these, but you MUST attribute them in your README file.
- You are also welcome to use our sound and image files. (space-invaders-assets.zip)
- You can use anything that you find on the web IF you know that it is licensed to allow for public use e.g. Creative Commons License. If you do this, you MUST attribute the URL and license in your README file.
- You can create them yourself!

Hints!

Here are some suggestions to get you started.

- Consider having multiple scenes: one for the introduction screen, one for level 1, one for level 2 and so on. The "SwitchScenes" sample demonstrates how to do this.
- You will want to animate things on the screen so that they move at a smooth and consistent rate (30-60 times per second is a good target rate). Avoid the temptation to have a single while... loop that does this. Instead, use a Java Timer or a JavaFX AnimationTimer to fire periodically and animate things on the screen e.g. moving the aliens, moving the player's ship, drawing a missile. The "SimpleAnimation" and "SimpleAnimationTimer" examples demonstrates how to do this.
- There are a large number of "magic numbers" that you will need to play around with, including player ship speed, alien speed, distance to move and so on. Consider defining constants at the top of your class, so that they're defined in one place. e.g.

```
// Speeds
public static final double PLAYER_SPEED = 3.0;
public static final double PLAYER_BULLET_SPEED = 6.0;
public static final double ENEMY_SPEED = 0.5;
public static final double ENEMY_VERTICAL_SPEED = 10.0;
public static final double ENEMY1_BULLET_SPEED = 4.0;
public static final double ENEMY2_BULLET_SPEED = 5.0;
public static final double ENEMY3_BULLET_SPEED = 6.0;
```

- Use a keyboard handler attached to the scene to capture keyboard input from the user. The "EventHandlers" samples demonstrate how to do this.
- Finally, look into some of the specialized classes in JavaFX for sound and graphics. For example, Media (https://openjfx.io/javadoc/11/javafx.media/javafx/scene/media/package-summary.html) and Image (https://openjfx.io/javadoc/11/javafx.graphics/javafx/scene/image/package-summary.html) classes may be useful.

Technical Requirements

- Your project should be implemented as a Gradle project in IntelliJ 2021. You should use Java 15 or later and Java FX 11. You may build and test your project under Windows, Linux or macOS.

- The default gradle tasks ("gradle build" and "gradle run") should build and execute your project. This will happen by default if you setup a new Gradle project.
- Your main class should be named "SpaceInvaders" and reside in a file named `SpaceInvaders.java'.
- You are free to use any sample code or examples provided in class. You may copy or modify this code freely, but attribute it in your README file.
- You may also use the image and sounds assets identified above, but you may not use source code from any other source without explicit written permission from the instructor.

Submission

Your directory structure for your assignment submission should be in subdirectories under your a2/ directory in your Git repository. It should use a standard file layout for an Gradle project. For example, it should **resemble** this configuration, with a build.gradle, app/src directory structure and so on (it does not need to match exactly!)

```
.
├── app
│   ├── build
│   ├── build.gradle
│   └── src
│       └── main
│           ├── java
│           │   ├── Enemy.java
│           │   ├── EnemyBullet.java
│           │   ├── Player.java
│           │   ├── PlayerBullet.java
│           │   └── SpaceInvaders.java
│           └── resources
│               ├── images
│               │   ├── bullet1.png
│               │   ├── bullet2.png
│               │   ├── bullet3.png
│               │   ├── enemy1.png
│               │   ├── enemy2.png
│               │   ├── enemy3.png
│               │   ├── logo.png
│               │   ├── player.png
│               │   └── player_bullet.png
│               └── sounds
│                   ├── explosion.wav
│                   ├── fastinvader1.wav
│                   ├── fastinvader2.wav
│                   ├── fastinvader3.wav
│                   ├── fastinvader4.wav
│                   ├── invaderkilled.wav
│                   └── shoot.wav
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
├── gradlew.bat
├── readme.md
└── settings.gradle
```

Your submission needs to include:

- All source code and resources required to build and execute your program.
- An Gradle + IntelliJ project that compiles everything using the specified JDK and Java FX versions, and which has a Run target that will execute your program.
- A `readme.md' file with any details required to help the TA grade your assignment.

Your readme file needs to include, at a minimum, your name, student number, the version of Java that you used (from `java -version'), and your operating system. If the TA needs to know anything else to run your assignment (e.g. undocumented hotkeys), include them in this file. For example:

```
Jeff Avery
12345678 j2avery
openjdk version "15.0.2" 2021-01-19
macOS 11.2.3 (MacBook Pro 2019)
```

Assessment

Your submission will be assessed roughly as follows:

5%
Complete submission. Code compiles and runs. Attribution in README file.
5%
When the application is launched, the game displays a Start screen containing required information and instructions for the user. The user can launch or quit the game from this screen.
10%
When the user starts the game from the Start screen, the game screen displays all required elements (level information, number of ships, game screen with aliens and player ship).
20%
Alien ships move across the screen (left-right) and descend periodically to threaten the players ship.
10%
Aliens fire missiles that can destroy the player's ship. If the player's ship is struck by a missile, it is destroyed and a new ship is spawned (if ships remain).
20%
The player can maneuver their ship using A/D keys and fire using SPACE. Missiles that are fired by the player destroy alien ships when they strike.
10%
When the level is cleared, the game progresses to a new level with faster movement. There are at least three levels (and 1/2/3 from the Start screen will launch them).
10%
When the game ends, the appropriate message will be displayed (i.e. "you win" or "you lose") and the player will be prompted to restart or quit the game.
10%
Compelling gameplay that includes graphics and sound effects!

Versions

1.0 April 14, 2021. Initial version.

1.1 June 1, 2021. Java 15 changed to Java 15 "or higher", to allow Java 16 builds.

1.2 June 3, 2021. Fixed small typos and uploaded to course website.

CS 349 User Interfaces (Spring 2021)

Cheriton School of Computer Science

University of Waterloo