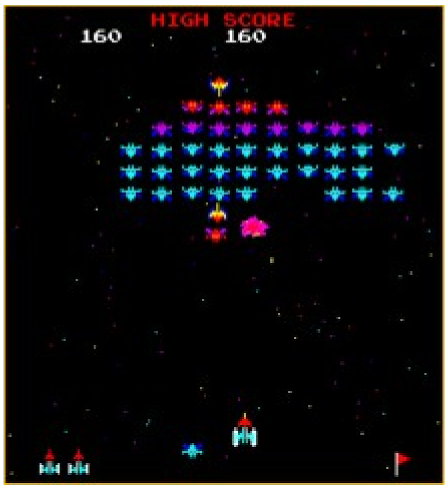
CISS 245: Advanced Programming
Galaxian Project Requirements (Subject to change)



The first thing to do is to google for “galaxian” and play the famous arcade game on the web for free. You can read up more about this game here: http://en.wikipedia.org/wiki/Galaxian.

As always, if the program crashes at any point in the game play, the project grade will be zero. If you do not turn in your project by the deadline, the grade is also zero.

Here are some project presentations in previous semesters:

- Andrew Woods
- Aastha Lamichchane
- Sushil Kharil
- Braydon Hampton
- Michael Fisher
- Caleb Fischer
- Darren Hays

REQUIREMENTS. Unless otherwise stated, every requirement is 1 point. There are two sets of requirements: software feature requirements and design requirements.

Feature requirements:

- When the program first runs, a welcome window is shown. The welcome window should show texts containing the name of your game and your name. The aliens scroll in from the right showing the points for each alien killed. Furthermore, it prints a message telling the user to press the spacebar to begin the game. If the user closes the program, the program halts.
- During the welcome or the game play, there’s a scrolling background star field.
- When the game begins, aliens are lined up near the top half of the window. There are different aliens. The aliens are not stationary but move left and right. However the images of the aliens do not overlap.
- The spaceship moves left and right using the left and right arrow keys. The image of the spaceship stays within the window.
- The spaceship can fire lasers. Each laser moves upwards. The lasers are separate by delays so that visually you can see distinct lasers. Each laser disappears on reaching the top of the window. The spaceship can fire lasers without waiting for the laser to disappear beyond the top of the window.

- When the spaceship's laser hits an alien, both the alien and laser disappears and an explosion occurs at the center of the alien's image. The explosion should last about 1 to 2 seconds.
- As the game play continues, aliens will randomly enter attack mode. When an alien enters its attack mode, it descends on the spaceship. You can choose any path for the alien. A straight line is fine. [See EXTRA CREDIT]
- Note that when the yellow color alien goes into attack mode, the red aliens in front of it (if any) will follow to protect it.
- During an attack mode, when an alien disappears from the window (either reaching the bottom of the window or passed the left/right side of the window during descend), it rejoins the alien formation.
- When the alien re-joins the formation, it flies from the top of the window and drifts slowly towards its previous position in the formation.
- When an alien is in attack mode, it also fires lasers. These lasers move downward.
- Alien lasers can collide with the spaceship. However alien lasers do not collide with aliens.
- When an alien is killed, points are awarded. The total score is shown at the top of the window. Different aliens are worth different number of points.
- When all aliens in a fleet are killed, a new fleet appears.
- The number of fleets killed is displayed at the bottom of the game window.
- A player starts off with 3 lives (i.e. 3 spaceships). Draw the remaining “lives” (i.e. spaceships) near the bottom right of the window. The spaceship that is currently under play (i.e. controlled by the user) should be moved slightly up and away from the status bar that shows the number of remaining lives.
- When the spaceship has collided with an alien or an alien laser, the spaceship “dies”, an explosion is shown. In other words you lose a life.
- If you have no more lives, a message indicating the end of game is displayed near the middle of the window. After a pause (say about 5 seconds), the program goes back to the welcome screen.
- The game must have sound. For instance when there's an explosion, a suitable explosion sound file must be played. You can also play some background music during demo, play, and end of game.
- The game should have roughly the same frame rate when run on different computers and the frame rate must be sufficient for the game to run smoothly.
- The game must not crash while running.

Design requirements: The above are the software feature requirements. The following are code design requirements. You should use as many classes together with inheritance and composition as possibly to simplify the structure of your software. Note that not all classes are mentioned below.

- There must be a class for the spaceship that the user controls.
- There must be a class for each type of alien.
- All alien classes must be subclassed from a parent alien class.
- There must be a class for lasers.
- There must be a class for stars (for the scrolling star field in the background).
- Classes must be kept in separate files (*.h and *.cpp).
- Proper names must be chosen.
- Code must be properly indented.

PRESENTATION

- You must present your work during the finals week (this is usually Thursday 9AM-12PM or after 1PM). Therefore you should aim to finish your project by Wednesday (of the finals week) at the latest and meet me in my office to do a demo. In fact you should probably finish most of your project the week before finals.

SUBMISSION

- You must turn in your project by Thursday 5PM of the final exam week. I have given you a demo folder containing sample code. You should use this folder. Just rename it. In particular, I must be able to execute your program after running “make” and “make r”. To submit your work, you must “tar” and “gzip” your directory. Make sure you remove any executable in your directory before submitting your work. For instance suppose your directory is called “p01” and “p01” is in your home directory. In your bash shell execute the following:

- cd ~
- cd p01
- make c
- cd ~
- tar -cvf p01.tar p01
- gzip p01.tar

This will produce p01.tar.gz. You submit your work by email p01.tar.gz to yliow@ccis.edu with a subject line of “ciss245 p01”. You must email using your college email account. Include yourself in your email so that you get a copy of the exact same email – this is for your own record.

EXTRA CREDIT

- During attack mode, an alien moves in a curve (not necessarily a straight line).
- Aliens group up together to form a strategy to corner the player's spaceship.
- The game prompts the player's name when it's a high score. It's up to you to decide how many high scores you want to keep – 3 is enough. The high scores with names are shown during the welcome sequence.
- You can implement variations. Many students successfully “graduating” out of my CISS245 usually have their own version. For instance Mark allow users to move vertically. Sam implemented a clone of Galaga (a sequel to Galaxian) where the aliens look more like bugs flap their wings and when you kill an alien, it breaks up into pieces and fall with gravity and the screen also shakes. Andrew has powerups that you can pick up so that your laser power is increased. Etc. Any variation is fine with me as long as it does not simplify the original galaxian project. There are many websites hosting 2D game development content such as font, images, sound bites, and music. Just google “free 2d game dev image sound asset”.

Suggestions/Advice

First of all, of course you need to study the demo folder I gave you. Create some simple exercises of your own. For instance make a scrolling hello world where the hello world scroll in from the right and disappear through the left side of the window. Draw a smiley face – your need to draw circles and probably lines. Make it blink or change its smile when a key is pressed. Use your imagination. This should not take more than 1 day.

You should build small prototypes of the game before collecting them together into a complete game. This is a general problem solving technique for large scale projects and is relevant to any type projects, software or otherwise. Here are some obvious prototypes/experiments:

1. Write a program that place your spaceship at the bottom along the y-axis and in the middle along the x-axis, i.e., in the middle of the bottom of the screen. This should be the same no matter what size your drawing surface was set to.
2. Building on the previous, allow the user to move the spaceship with the left and right arrow keys. Make sure that when the spaceship stays in the window. In other words, if the spaceship has reached the left side of the window, pressing the left arrow key won't do anything to the spaceship.
3. Building on the previous, now when you press the spacebar, a red rect representing a laser appears on top and in the middle of the spaceship. When you let go of the spacebar, the red rect disappears. Make sure that it works even when the spaceship is moved to a different position.
4. Building on the previous, now when you press the spacebar, the laser appears and then flies on its own vertically up on its own. Just work on one laser. When you press the spacebar, the laser that is in-flight is reset to the top middle of the spaceship again. So there's only one laser. [HINT: You of course needs to have variables for the position of the laser. But you also need to have a boolean flag to tell you if the laser is in flight or not, I.e, whether the laser is alive or not. When the player presses the spacebar, you set the alive flag for the laser to true. In the section of code that moves game objects, you move the laser only when the laser is alive. In the section of code that draws everything thing, you only draw the laser only if the alive flag of the laser is true.,]
5. Build on the previous so that you can see more than one laser. So I can fire a laser, move to another position and fire another and I'll see two lasers in flight. [HINT: You need to have an array lasers. When a player pressed the spacebar, you run through the lazers and find one that is available, i.e., not in flight, i.e., the one with the alive flag having false value. Use that particular lazer for this spacebar key press.]
6. Write a program so that an alien moves left and right on its own.
7. Building on the previous, write a program so that an alien moves left and right (when it's in the formation mode) and it randomly goes into attack mode and decides on fly down in a straight line.
8. Building on the previous, write a program so that an alien moves left and right and it randomly decides on fly down in a straight line and when it reaches the bottom, it reappears at the top and drift back to its original position. To test this, you will need to draw a rectangle that moves with the alien is in formation mode and even when the alien goes into attack mode and flies down, the rect is still moving left-and-right constantly on its own. That way, you can check that when the alien rejoins the “fleet”, it goes to its place in the formation.
9. Now add some collision detection, i.e., when the laser touches the alien, make the alien explode. I suggest making the explosion simple – for instance just make the alien disappears. Instead of building on top of the previous program, you might want to build a collision detection prototype where the alien is stationary. To make sure that the collision detection works for with the alien at different places, you can for instance make the alien appears at different places randomly when the program runs. [HINT: You probably want to have a boolean flag, alive, for the alien.]
10. You make a whole fleet of aliens. Make them move together.
11. Now include score. At this point, you have roughly a game.
12. The difficulty is with the “attack group”, i.e., 3 red aliens and the yellow alien attacks in a

group.

13. After that you want to include lives. A player starts with 3 lives.
14. Don't forget to increase the rate of attack as you increase the level.
15. Don't forget the welcome screen.

I'll leave the rest to you. If you follow the above steps, you would already have a pretty good grasp of the project as a whole. The above is only a suggestion on project breakdown – I'm not saying you have to follow the above experiments religiously. There are many ways to breakdown a large task into smaller ones – you can do it your way.

Constantly clean up your code. As much as possible use constant variables instead of hardcoding constants. I'll be giving guidelines/hints/suggestions to help you along the way. It's also a good idea when writing a game to keep a paper folder with sketches/drawing of your ideas. The process of game development is actually very similar to for instance a movie production. So keep sketches of the “storyboard” and sequence of pictures of what the player sees.