# Description:

Using Python’s Pygame (game engine library), Chris and I have recreated the original arcade game Asteroids, to the best of our abilities using vector graphics. Unfortunately, due to time constraints, our knowledge of Python and Pygame, and task brief constraints, the player’s Spaceship is not a drawn isosceles triangle like in the original game, and the voxel effect.

The main purpose of the game is to survive a barrage of asteroids by destroying them, thus earning points, as well as surviving and destroying randomly spawning UFOs’ that will fire bullets to destroy you. The points that you earn from destroying asteroids and the UFOs’ will be displayed if and when you die on the game over screen. Above all else, the main purpose of this game is for you to have fun and experience the popular arcade game on your laptop.

# Prerequisites:

To play this game efficiently the recommended device specifications are:

* A Mac or Windows OS.
* At least 4GB RAM.
* “x” Amount of Storage.
* For the code itself: Python, Pygame, Math Lib, Random Lib, OS Lib.
* A low-end laptop or PC (minimum) with a functioning keyboard and mouse/trackpad.

To create the game, the requirements are:

* An IDE such as Visual Studio Code, PyCharm, etc…
* Python, Pygame, Math, Random, OS Libraries.

# Installation/How to Play:

To play the game, you just need to run the executable file (.exe for Windows, .app for MacOS), and the game will load without any problems (unfortunately the conversion to .exe and .app doesn’t work, you will have to use the alternative installation method). Alternatively, if you have an IDE and all the required libraries installed, open the program’s file, and run the Asteroids.py script and the game will load.

To play the game itself (after running/installing), in the main menu there is a button called instructions. If you click on that with your cursor a screen will appear showing you the aim of the game, and the key binds to functionally play the game. Further there is a README file that contains the key binds with their functions, and the aim of the game.

# Bugs & Errors:

In the game, unfortunately there are a few annoying bugs that can impact gameplay. The most significant bug is when the player collides with the edges of the screen window. When this happens, the player can rotate in every direction, but cannot escape the corners/edges. To mitigate this bug, you will have to restart the game entirely. Another annoying bug is the in-game pause menu transition to the options screen and back. Entering the in-game options screen is simple and straightforward with no errors, however when you return to the pause menu, the options screen is still visible despite no longer being active. You can also access and press the buttons specific to the options screen in this state.

# Challenges & Successes:

During the game development process, Chris and I encountered many challenges before finally developing a functioning game. The challenges we encountered were mainly in the programming phase, as the design phase was reasonable straightforward. The challenges we encountered were mainly; logic errors in the code, rendering issues, syntax errors (which were the most tedious of errors that caused us “hell”), runtime errors, and math errors.

Despite the challenges, we managed to achieve many successes throughout the two-month period. We managed to draw all the sprites by hand (that look like the original game) and fulfilling our limitation, successfully implement game-states with a state manager to control the flow of the game, create and implement player functions that are nearly identical to the original arcade game, and create the audio using Beepbox (with some level of struggle).

# Game Mechanics:

The main mechanics that the game features are:

* **Player Controls:** Using WASD, you can rotate 360 degrees, and travel in the direction facing.
  + **Rotation:** The player can rotate 360 degrees.
  + **Thrust:** The spaceship can accelerate in the direction its facing.
  + **Shooting:** The player can shoot bullets in the direction facing to destroy the asteroids and other entities.
* **Asteroids:** The primary obstacle in the game.
  + **Breakable:** All the asteroids can be destroyed by the player to earn points.
  + **Movement:** Asteroids drift in various random directions and speeds.
* **Enemies:** Flying Saucers that appear periodically.
  + **Movement & Shooting:** Saucers can shoot at the player’s spaceship.
* **Screen Wrapping:** Objects that move off one edge of the screen reappear on the opposite edge.
* **Score & Lives:** The player earns points by destroying asteroids and saucers but loose lives when colliding with them.

# Game Dev Limitations:

The limitations of developing our game are:

* **Demoscene (Esoteric):** Game works on an obscure platform. The game in future phases will be able to work on older devices (as well as cross platform), with the aim of being able to run on the Nintendo Switch, Nintendo DS, and possibly the CASIO Classpad.
* **Graphics (Illustrator):** Draw all your art by hand. Using Piskel and other art tools, to draw the graphics. With the exception of fonts.
* **Audio (Beep Bloop):** Audio generated by gameplay. Using beepbox for the sound effects.
* **Wildcard (Window Dressing):** Game windows are part of the gameplay. In our case screen wrapping is the key window feature.

# Planning (Gantt Chart):

# A screenshot of a video game Description automatically generatedA screen shot of a computer Description automatically generatedIn Game Screenshots:A screenshot of a video game Description automatically generated

# References/Bibliography:

McFadden, A. (2021) *Asteroids Disassembly* , *Asteroids disassembly*. Available at: https://6502disassembly.com/va-asteroids/ (Accessed: 27 June 2024).

*Chatgpt* (no date) *ChatGPT*. Available at: https://openai.com/chatgpt/ (Accessed: 27 June 2024).

*Microsoft copilot: Your everyday ai companion* (no date) *Microsoft Copilot: Your everyday AI companion*. Available at: https://copilot.microsoft.com/ (Accessed: 27 June 2024).

*Asteroids 1979 - arcade gameplay* (2015) *YouTube*. Available at: https://www.youtube.com/watch?v=BgloG8yt-jA (Accessed: 27 June 2024).

*Pygame Documentation pygame.draw* (no date) *Pygame.draw - pygame v2.6.0 documentation*. Available at: https://www.pygame.org/docs/ref/draw.html (Accessed: 27 June 2024).