A Case Study: Programming Arcade-Style Games Using a High Level Programming Language

Apoorva Anand[1], G Janakiram[2]

[1]Under-Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India
[2]Under-Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India

ABSTRACT

The project aims to implement classic arcade style gameplay with a modern flourish of weaponry and techniques. The game is reminiscent of elements from games like Pong and Agar.io.

Using the module pygame as the backbone of the GUI, we hope to achieve satisfying replayability coupled with an introduction to game design and game designing concepts. The project is a crude attempt at grasping collision theory and 2D dynamics. Eventual expansion is possible as we use object oriented programming that emphasizes modularity and inheritance, concepts that'll help the game to always stay as a "work-in-progress", something we as a team greatly support due to the infinite possibilities it brings with it.

Keyword: - *Game, game design, 2D dynamics, modularity, inheritance*

1. Objective of the Game

The objective of the game is to destroy the various kinds of blobs bouncing around the screen with the help of projectiles released by a launcher with the aid of power-ups.
Major cues are taken from famous arcade titles like Arkanoid, Pong and Asteroids.

E.g. 1: As per Pong, the concept of a "paddle" is seen in the "launcher".
E.g. 2: As per Arkanoid, "enemy units" are seen in the form of blocks.

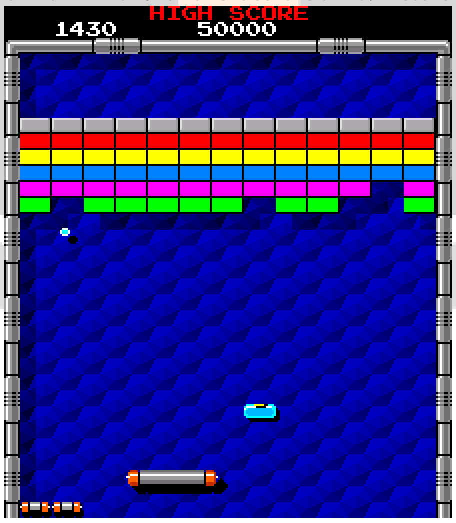
1.1 Modules to be Used in the Game

- Blob collision with wall: Takes care of blobs that hit the wall and modify movement according to their type
- Blob collision with other balls: Takes care of blobs that hit each other and modify movement according to their type
- Blob collision with projectile: Elimination or splitting of blobs when hit with a projectile
- Projectile movement: Dictates projectile movement
- Launcher module: The only "character" controllable by the user
- Projectile power ups: Give the player abilities to help eliminate blobs
    - Arc: Releases projectiles in an arc
    - Slow: Slows down all the blobs
    - Destroyer: Destroys all blobs (of every type) in path
- Blob: A circle that does nothing but bounce around
    - Bouncy: Speeds up every time it hits the walls
    - Spiky: Releases its own projectiles (spikes) at certain time intervals or when hit by the user
    - Rubbery: Makes any projectile that hits it, rebound to source
    - Time: Has to be destroyed before the timer ends or the game ends
    - Splitter: When hit, splits into two smaller blobs that can split once more before getting destroyed

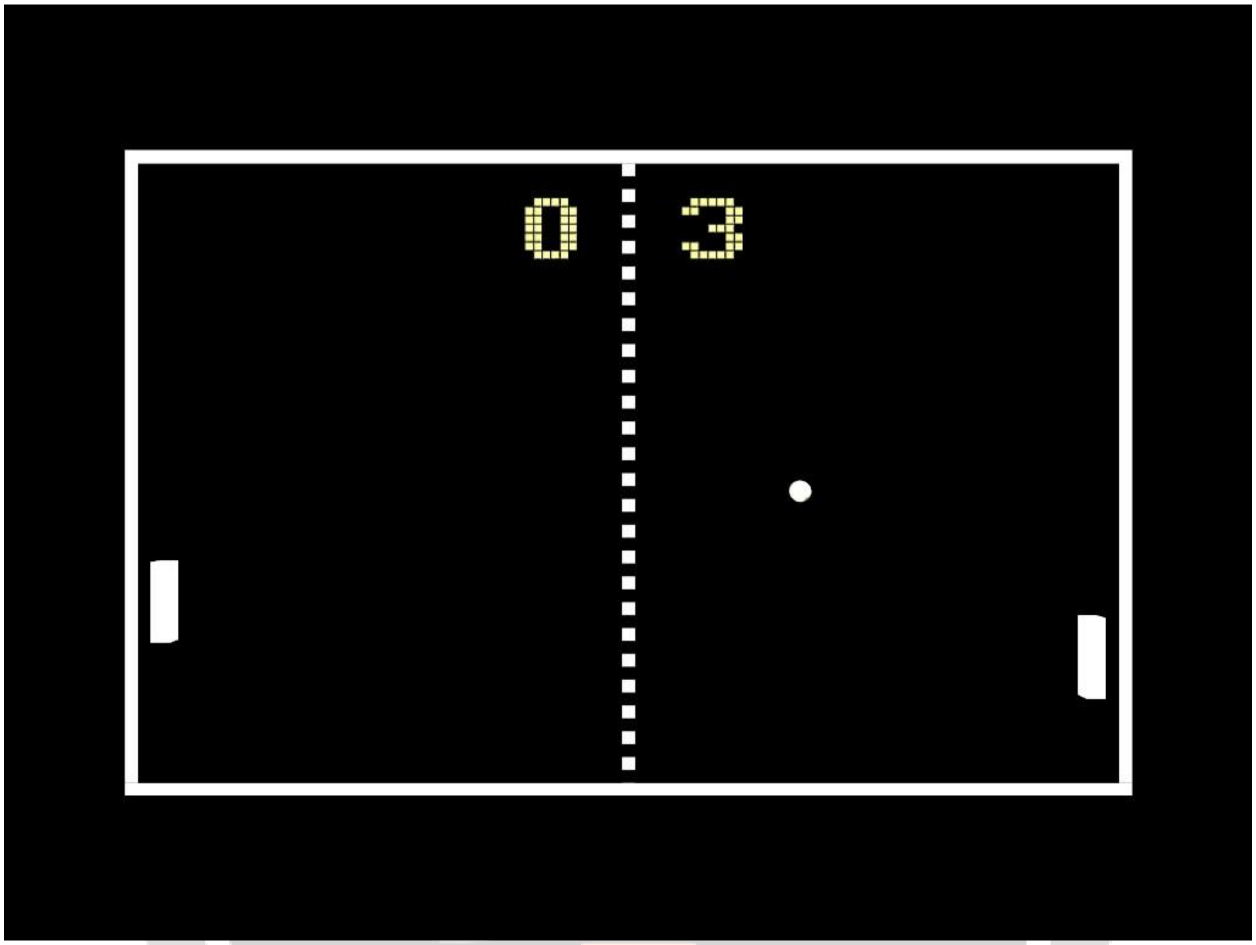
2. Existing Systems

The beautiful thing about games is that you don't need to have a "better system". Games can coexist. The only thing required from a game is that it be fun. And of course, there are a number of existing games that are fun. In fact, we've taken inspiration from them! Specifically, the games are – Pong, Arkanoid and Asteroids.

The Main Menu of a Classic Game: Asteroids



The Gameplay of a Classic Game: Arkanoid



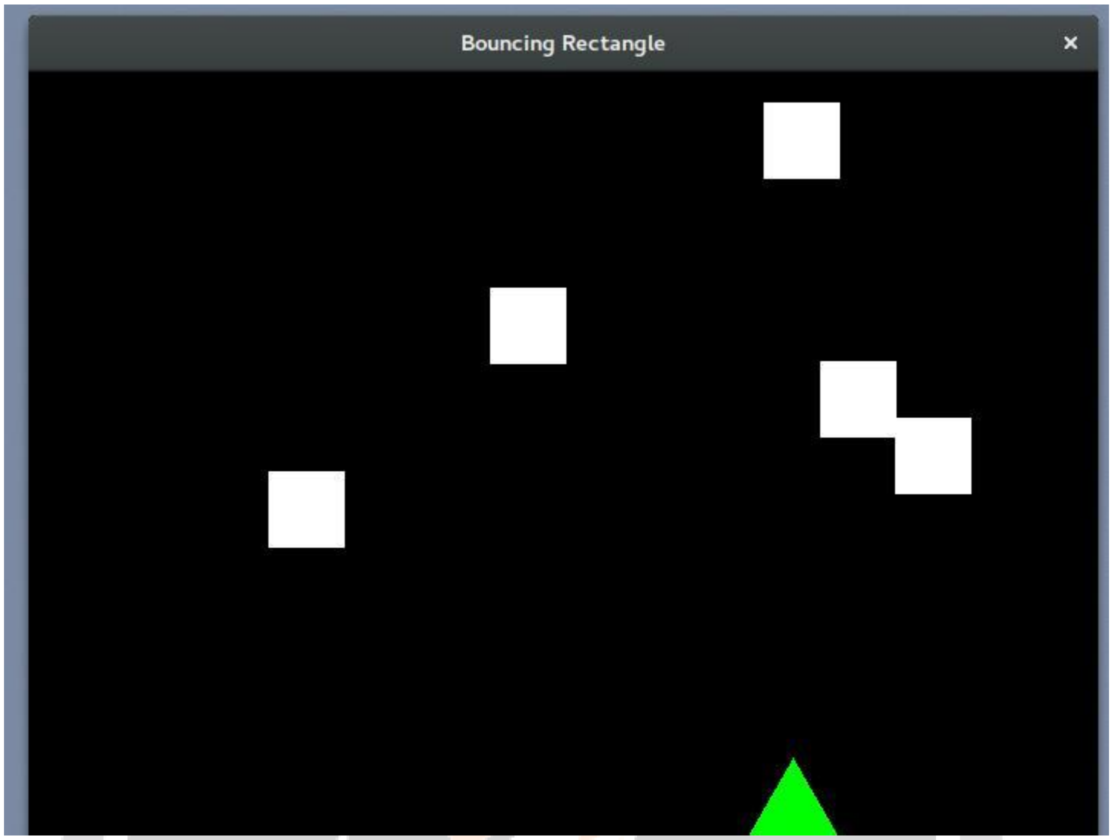
The Gameplay of Another Classic Game: Pong

2.1 Drawbacks of Existing Systems

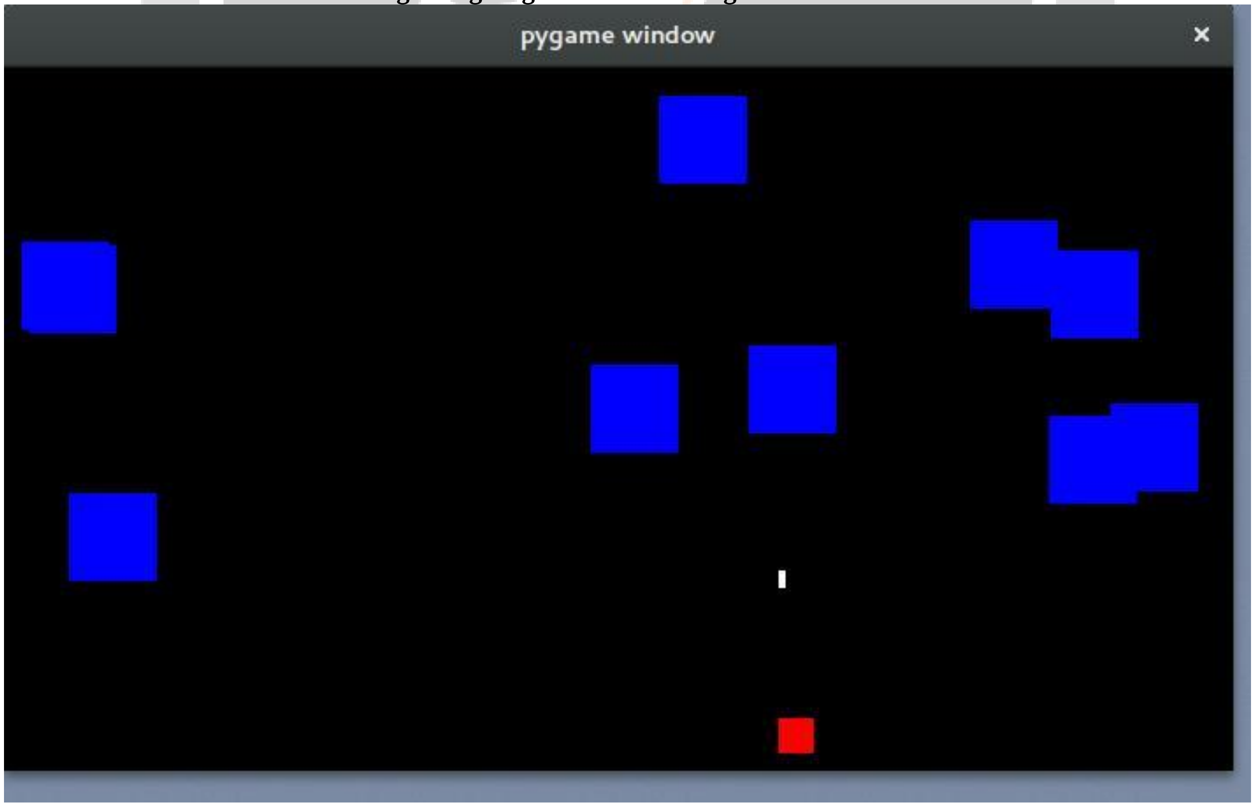
These old arcade games were limited by the kind of technology and hardware that existed then. Cool power-ups couldn't exist – We didn't have enough memory for them! There's been a huge increase in community support for game development, which enables us to make an "arcade" style game with "modern" capabilities.

2.2 Proposed System

Our game will feature a number of blobs bouncing around the screen, each with their own properties. The objective of the game is to destroy all of these blobs using a launcher that shoots projectiles. It is possible to "power-up" the launcher so it releases special kinds of projectiles with properties of their own. All this is made possible with the massive increase in available memory and computational power.



The beginning stages of our arcade game: Block World



The gamplay of our game: Block World

3. Object Oriented Analysis Diagrams

These following diagrams illustrate the game and its components in a lucid way so that we can get a "bird's eye view" of the entire process.

3.1 Use Case Diagram

The use case diagram helps us see the reach of the proposed system as it interacts with the player and what the player can access or modify. The use case diagram allows us to dedicate our attention to the components that matter instead of the periphery and clutter.

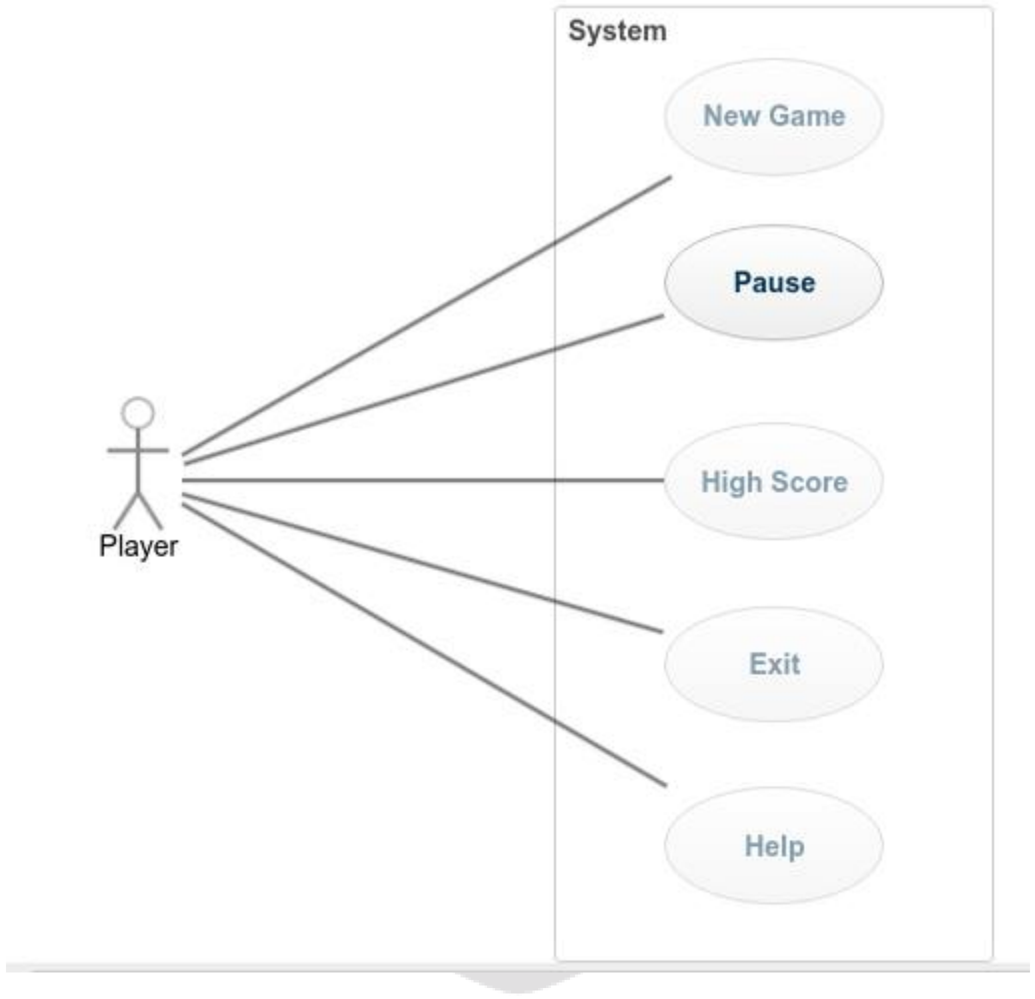


Fig-1: Use Case Diagram

3.1 Class Diagram

The class diagram gives us a very good look at the technical side of things as the more computer science related parts are revealed in it. The look and feel of the system becomes apparent in this diagram.

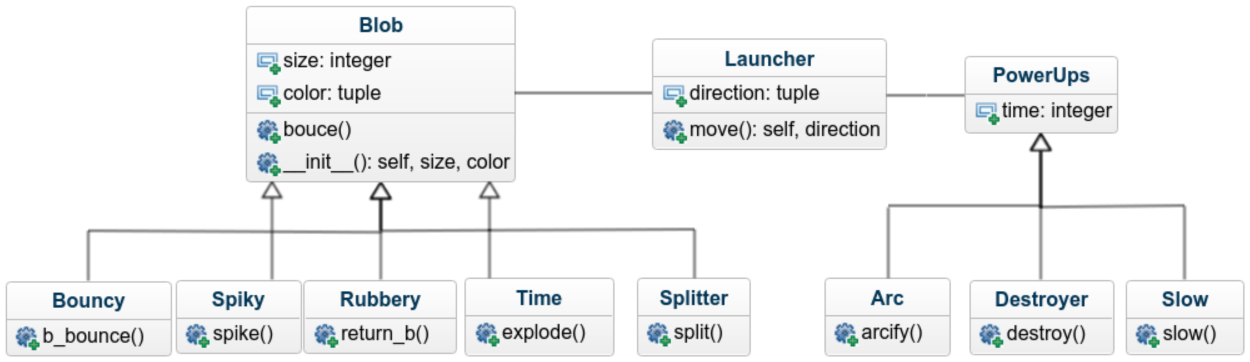


Fig-2: Class Diagram

4. CONCLUSIONS

The game we have designed makes an attempt at understanding the nuances of game design during its building and helps us appreciate the effort put in by game developers at large. The limited resources and how they have been thoroughly used by the developers of the arcade games helps us appreciate the sheer can-do attitude of these great minds.

Complex computer and video games may provide a vehicle, based on appropriate theoretical concepts, to transform the educational landscape.

Video games have gained high popularity nowadays that researchers have taken an interest in its use as an educational medium.

5. ACKNOWLEDGEMENT(S)

Mr. S Jaya Kumar[1]
Mrs. E Saraswathi[2]

[1]. A.P(S. G), M. Tech. Computer Science and Engineering, SRM University, Ramapuram, Chennai
[2]. A.P(O. G), M. E. Computer Science and Engineering, SRM University, Ramapuram, Chennai

6. REFERENCES

http://ieeexplore.ieee.org/document/5529699/ [1]
http://ieeexplore.ieee.org/document/6624223/ [2]
https://link.springer.com/article/10.1007/s11423-006-9001-x [3]

[1]. Implementation of a digital game-based learning environment for elementary education
[2]. Choosing the infrastructure for entertainment and serious computer games – a white room benchmark for game engine selection
[3]. Game object model version II: a theoretical framework for educational game development