METEORITES

Project Report

TTCJ Software Ltd. (Group 27)
By: Chanokpol Janveerawat & Keison Tang

Introduction

This report highlights our progress on our project thus far. The request from the client was to create a game similar to Atari 2600's 'Warlords'. We have included details on the development of our game, including tools, design pattern used, and any issues we encountered along the way. Since the game is to be played by a twelve-year-old and his friends, all elements of our game were designed to suit players around this age group.

Client Requirements

Objective : A 'Warlords'-like game that is both enjoyable and suitable for twelve-year-old players.

We have made considerable progress towards meeting the specifications set by our client. In Meteorites, we have implemented several key gameplay elements from 'Warlords' although with our own twists and theme. When the game is executed / run, a main menu greets the user, where they can choose an option from either single player, multiplayer (two players) and exit (***See Appendix 1***). In 'Single Player' mode, a single user controls one paddle using the keyboard and plays against three AI controlled paddles. In 'Two Player' mode, two paddles are controlled by the two users (or by a single user for those who want a challenge) and play against two AI. The input controls in single player are the left and right arrow keys. In two player mode, the controls are 'A' and 'D' for the left hand side paddle and the left and right arrow keys for the right hand side paddle. We have chosen these keys as they are both user intuitive and widely used game control configurations for movement.

When the user selects a game mode using the mouse, the in-game screen is displayed and, a three second countdown timer is displayed in the middle of the screen. During the countdown period, no objects are able to move and any user inputs are rejected. The paddles are set in their starting positions and the meteorite is placed in the middle (***See Appendix 2***). At the end of the countdown, the meteorite starts moving at a set speed in a random direction and the paddles are free to move. A two minute timer is displayed and starts counting down when the game starts.

The meteorite is able to destroy planets' rings which are made up of orbiting ice when it collides with them, and the meteorite is deflected when it contacts a paddle or the boundaries of the game window. Appropriate sounds effects are played to indicate the ball has made contact with something. The game progresses as bases (planets) and walls (ice cubes) are destroyed (***See Appendix 3***). Depending on the game mode, different win conditions will be checked. At anytime during the game, before timeout, the player can press 'P' on the keyboard to pause the game, or 'ESC' to get the option to quit the current game mode.

Win condition for Single Player mode

Before the 2 minute time-up

- The player wins if all other (AIs) planets are destroyed.
- The player loses as soon as the player's planet is destroyed.

After the 2 minute time-up

- If the player has the most ice cubes remaining, the player wins
- If the player has the most equal number of ice cubes remaining, the game is a draw.
- The player loses if they have the least number of ice cubes remaining.

Win condition for Two player mode

Before the 2 minute time-up

- If both players' bases are destroyed, the game is over and the AI wins the game.
- Otherwise the last player/AI standing, wins the game.

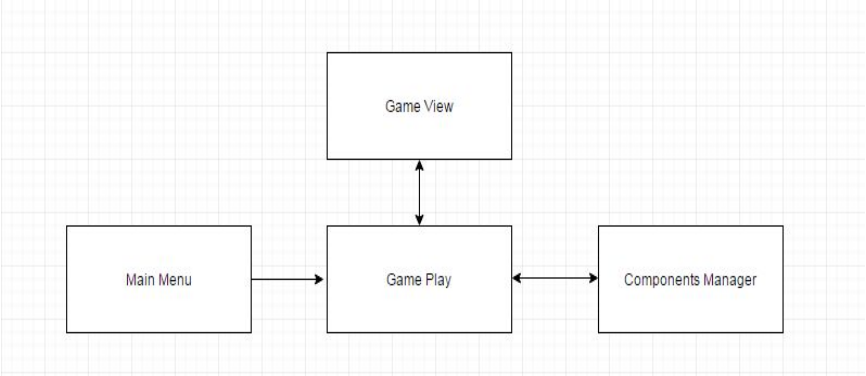
After the 2 minute countdown

- The player/AI that has the most number of ice cubes wins. A draw is possible if multiple players have the same, (and highest) number of ice cubes left.

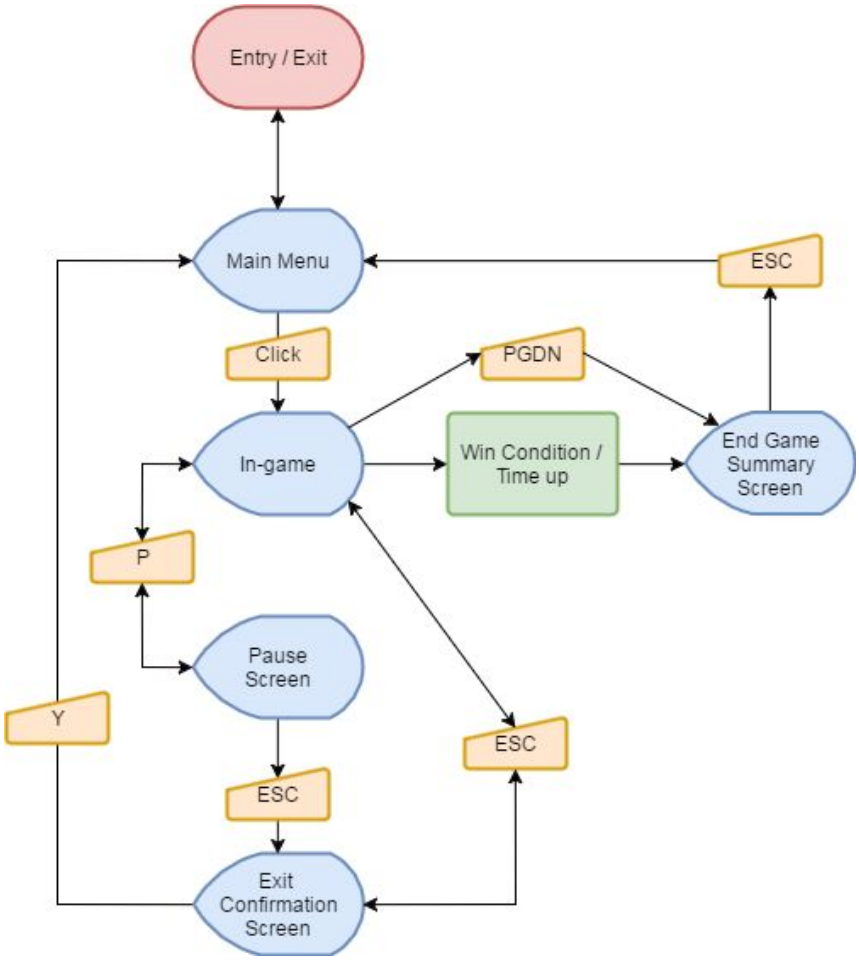
Once the game is over, there is a message that tells the user if they have won or lost, and the player can return to the main menu by pressing ESC key (***See Appendix 4 & 5***).

System Diagram

A high level system overview of our project is shown on the right hand side. A game play instance is created, with relevant game components by the Component Manager, depending on the selected game mode. User input to the game as well as the rendering of game's content is handled via the Game View.



The game navigation system showing how the user is able to access and navigate to each of the game scenes is shown below.



Development Issues

- Implementation of AI
    - Due to time constraints, we had to change paddles' path from the prototype to make the implementation of AI easier. Initially, we were going to store all the ball's future position in an array and find the point of intersection between them and the AI's path. It was very hard to come up with an accurate way of doing this since the ball can travel with different velocity and the intersection might not occur. The way we solved this problem was by using the equation of a line y=mx+c. First, find the meteorite path's gradient by using its velocity and then find the C value. X and Y values can then be easily determined. For example, to find whether the meteorite is traveling to the AI's vertical defense path, we substitute the X value (which is fixed) into the equation and see if the Y value

falls in the range of the AI paddle's path. If it does, move the AI's paddle to that Y coordinate.

- Git
    - Initially we were not aware of the branch feature on git. All changes were made on the master branch and this caused conflicts when we worked on different features and tried to merge them. Working on the same branch also made it difficult to go back to the previous version of the code when something went wrong.

Extra Features

To make the game more interesting and fun to play, we have implemented additional features to the game as follows:

- **AI is made to be imperfect**. AI controlled paddles will only respond to the ball when the ball is traveling in a particular direction. A ghost ball (invisible to the player) is also implemented to confuse the AI. The AI will try to defend its base from both the actual and ghost meteorites.

- **Ball's speed increases over time.** Ball velocity increases by a small amount every time it hits a paddle. This makes the game more fun to play as the player progress toward the end of the time limit.

- **Powerups.** We have implemented two collectable items in the game:
    - **10 second time decrement.** When the meteorite collides with this collectable, 10 seconds will be deducted from the 2 minute timer.
    - **Snowflake** - All paddles, including the AIs', will be frozen for 5 seconds.

- **Ghost Paddles.** This feature allows paddles to be free to move and deflect the ball even when the base associated with it is destroyed. This allows for more exciting gameplay, especially when it comes to Two Player mode and one of the human player's base is destroyed. This allows them to still be able to play and interact with the game instead of just watching the other player and waiting for the game to finish.

- **Countdown voice, game end sounds.** This feature increases the feedback and sensory experience of the game to a player. These were included to accompany the visual cues in the game.
    - **Countdown voice.** A voice counts down from three, two, one in parallel with the countdown timer on the screen.
    - **Win / Lose jingle.** Either sound is played at the end of the game depending on if the human player has won or lost.

Development Tools

Java - Object orientated programming language is suitable for this project. We chose Java because it is an OOP language and is platform independent. This means our product will be able to run on any computer and on any operating system. It also has JavaFX which is a GUI library which is what we used to render the game objects on the screen.

Bitbucket was the version control service we used. We chose Bitbucket as it allowed us to create private repositories. It acts as an online store for all of our code and allows multiple people to work on the project at the same time without the hassle of having to save and share multiples updated files between us. It also tracks all changes everyone has made throughout the project. Branch feature of git allows us to work on different features at the same time.

Software Design

Since there were many repeated objects in the game such as paddles and walls, using OOP was necessary to reduce code repetition. OOP also encourages encapsulation, all fields can only be modified through pre-defined methods. This ensures that the fields of an object cannot be misused or manipulated by external code.

We addressed cohesion issues by making game components with their own getter and setter methods to modify and access their fields. All methods within a class are related to each other. Although inheritance introduces high coupling, it was used as a way to reduce code repetition for common game properties that are unlikely to change during the development.

Classes can be categorised into 3 categories: Model, View and Controller. In our case Ball, Brick, Bat etc. are our models. These classes only store data that are relevant to themselves. The stored data can be accessed by calling getter and setter methods using a controller class. The controller also controls what needs to be render on the screen by calling methods in View object. View classes handle user's input as well.

Software Development

Agile - We planned, coded and tested the game iteratively; to avoid unforeseen problems or changes that might occur close to the end of the project. We chose Scrum; which is one of the Agile Frameworks, to assist with the development of this software. This included things like writing up user stories, using an online scrum board, and pair programming. Scrum board made

it easier for us to keep track with our progress during the project. By putting our progress on the board, it was easy to see what needed to be done, what was being carried out, what was blocked, and what was completed. Tasks were assigned to each of us using the board. This was done to make sure that we wouldn't be working on the same task, thus avoiding conflicts. User stories helped us tackle the project feature-by-feature. Important stories were addressed first then followed by least important ones. A screenshot of our scrum board can be seen in **Appendix X.**

Future Improvements

To improve the functionality and player experience the following features could be implemented:

Include game instructions

- Our game does not have instructions on which keys to press to control the player's paddle. Therefore, it might be confusing for new players.

Better collision detection / Game physics

- The meteorite in our game only deflects in the horizontal direction. This causes the meteorite to go through the paddle when it hits the paddle from above and below.

Implement High score system

- The winner will be able to enter his/her name and have it recorded in the game. This was our planned feature but due to time constraints we did not get to implement this in our game.

Use of Threading / More efficient code

- Although we have tried to reduce code repetition by separating it into functions and different classes, the main game controller class is still very large. The game also becomes laggy sometimes. We think this issue will be solved if we implement the game to run on multiple threads and write the code in a more efficient way.

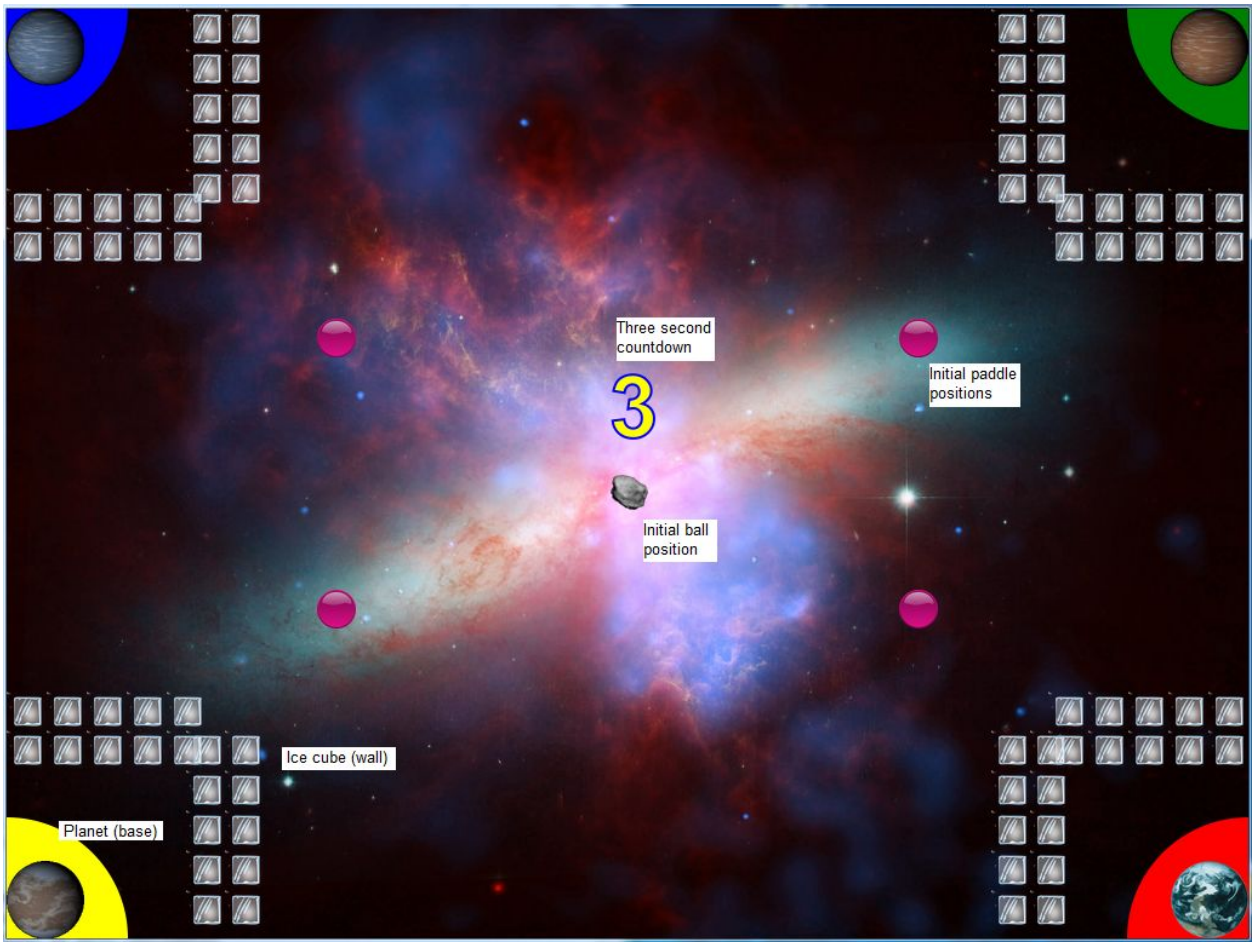
Conclusion

Given the time constraints set by the client, we still managed to develop and implement a game that met all the requirements given along with a few extra features that we felt were necessary to improve the gameplay experience and functionality of the system. Although there was much more room for improvement, we believe the project is satisfactory to the client's needs. Given that this was the first project for our company, along with the given timeframe, the project can be deemed to be a success. In any future projects, we can apply the valuable lessons and experience we have gained from the development of 'Meteorites'.

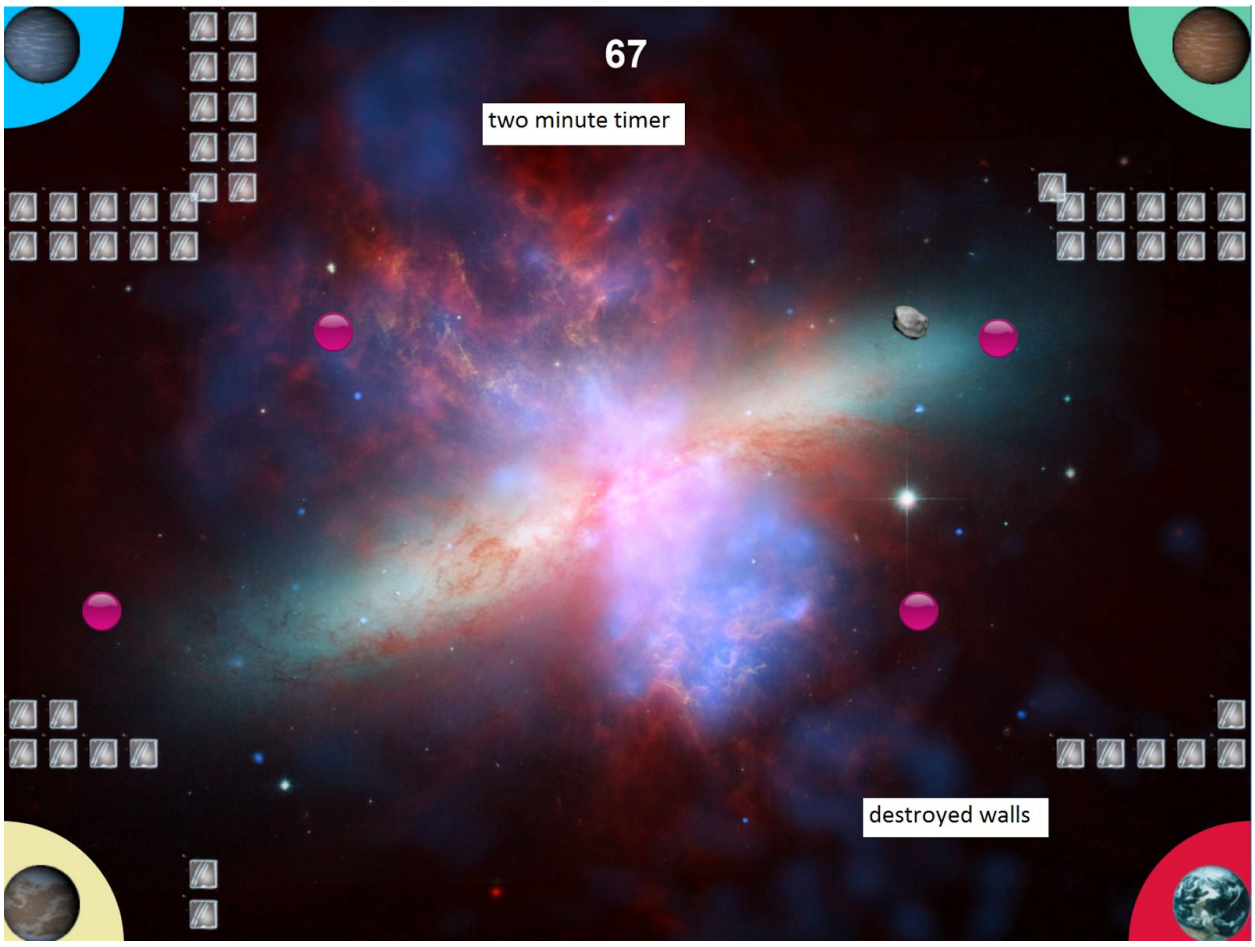
Appendices

Appendix 1 : Main Menu Screen

Appendix 2 : Still image showing the moment a user enters a game mode

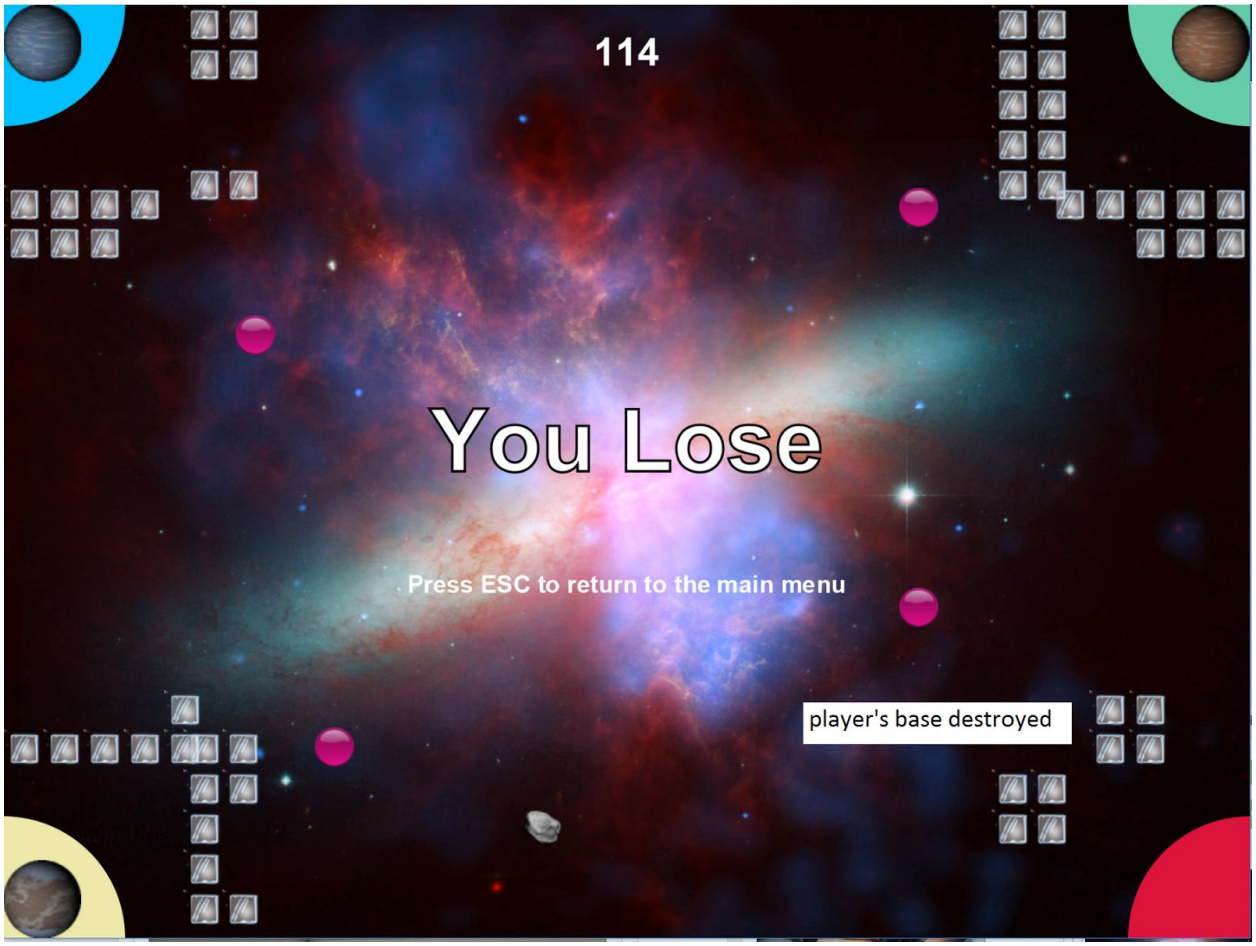


Appendix 3 : Game in progress



Appendix 4 : Win screen

Appendix 5 : Lose Screen



Acknowledgements / Credits

Sources for sounds used in game.

Countdown Voice
https://www.freesound.org/people/killkhan/sounds/150206/
Game Start Sound
https://www.freesound.org/people/FartBiscuit1700/sounds/368691/
Base Hit Sound
https://www.freesound.org/people/timgormly/sounds/170144/
Wall Hit Sound
https://www.freesound.org/people/coby12388/sounds/222575/
Paddle Deflect Sound
https://www.freesound.org/people/fins/sounds/146726/

All sounds used under the CC Licence.
https://creativecommons.org/licenses/by-nc/3.0/

Sources for images / sprites.

Ball Sprite by ‘phaelax’ --CC Licence
http://opengameart.org/content/asteroids

In-game background image - free to use
http://more-sky.com/WDF-326448.html