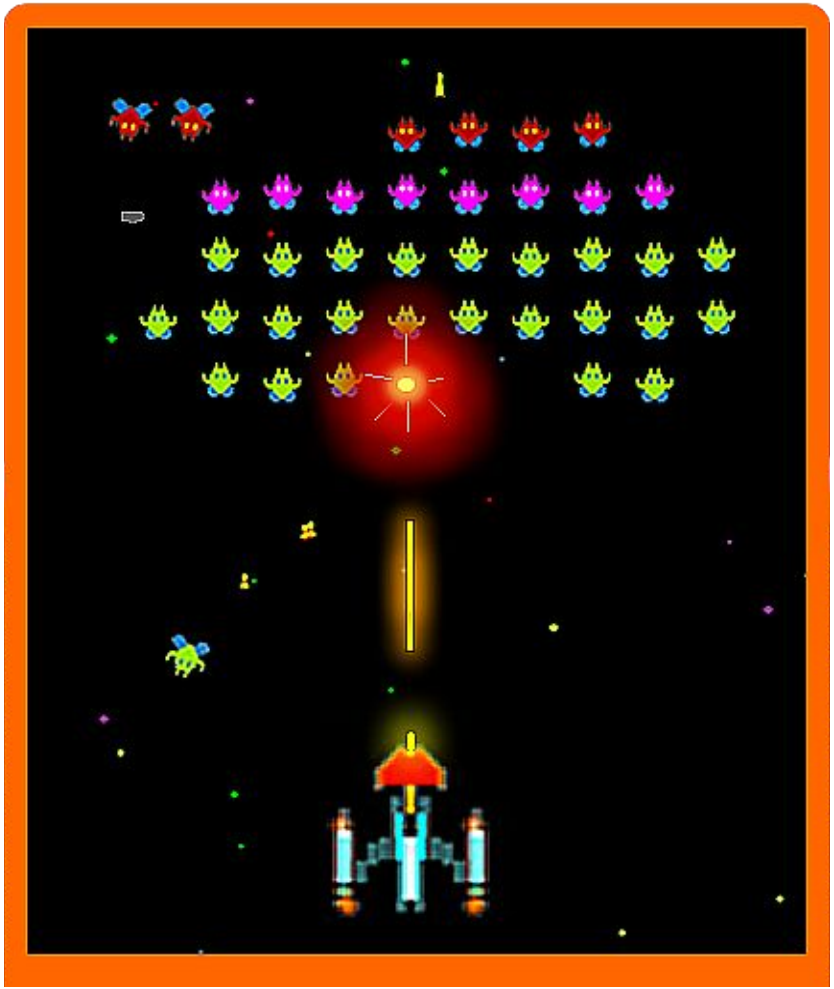
Game Development

Space Shooter

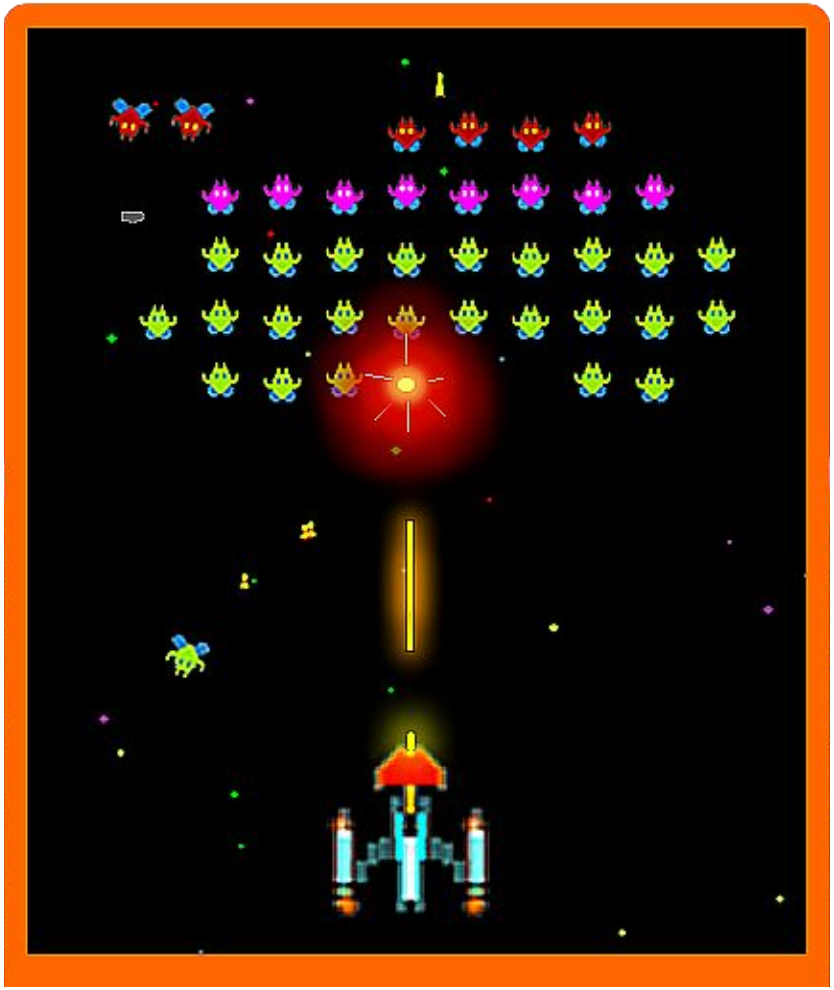
Space Shooter is a maze-based 2D game which was developed by Tomohiro Nishikado and first released in Japan in 1978.



Characters

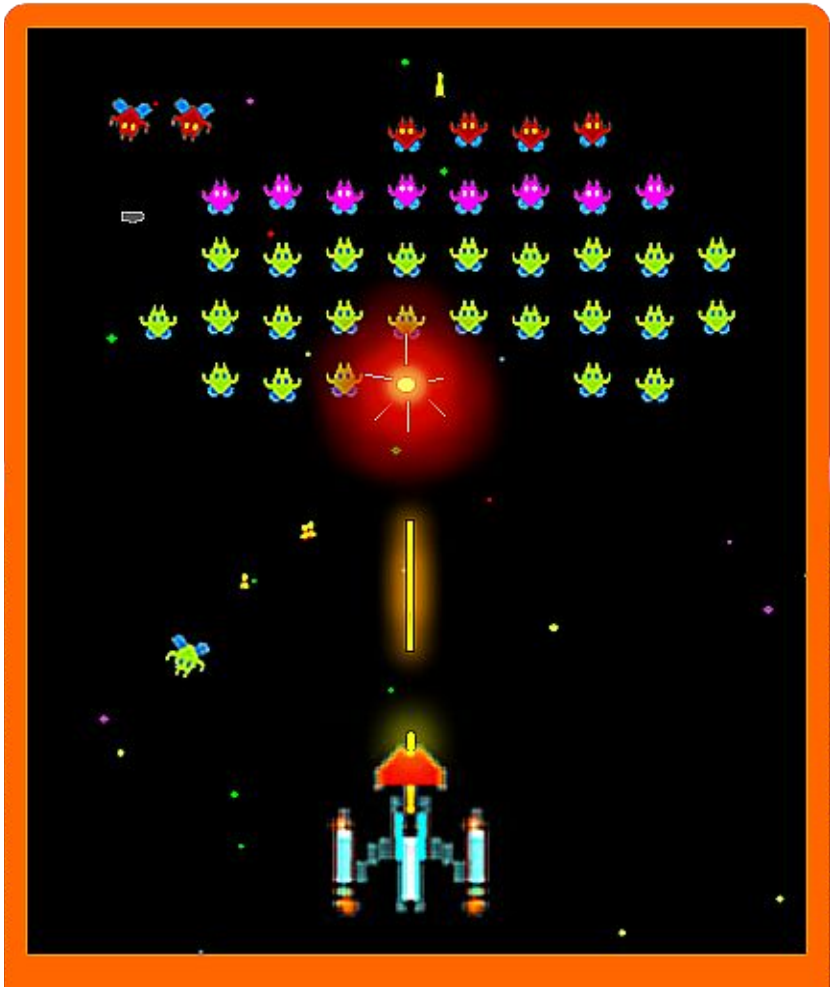
There are a following characters in the Space Shooter Game.

1 Space Invader (Player).
Multiple Enemy Ships.



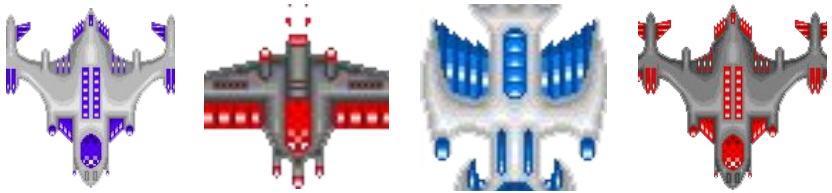
Characters: Invader

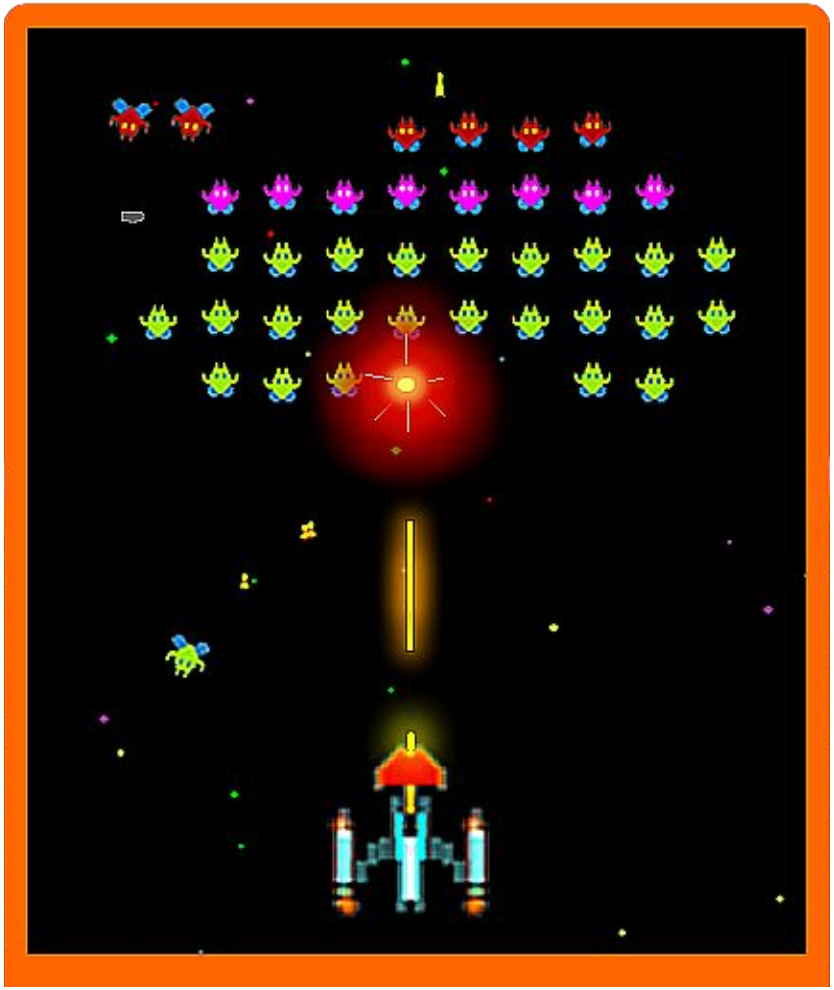
The Space Invader is controlled by the player with the help of arrow keys.



Characters: Enemy

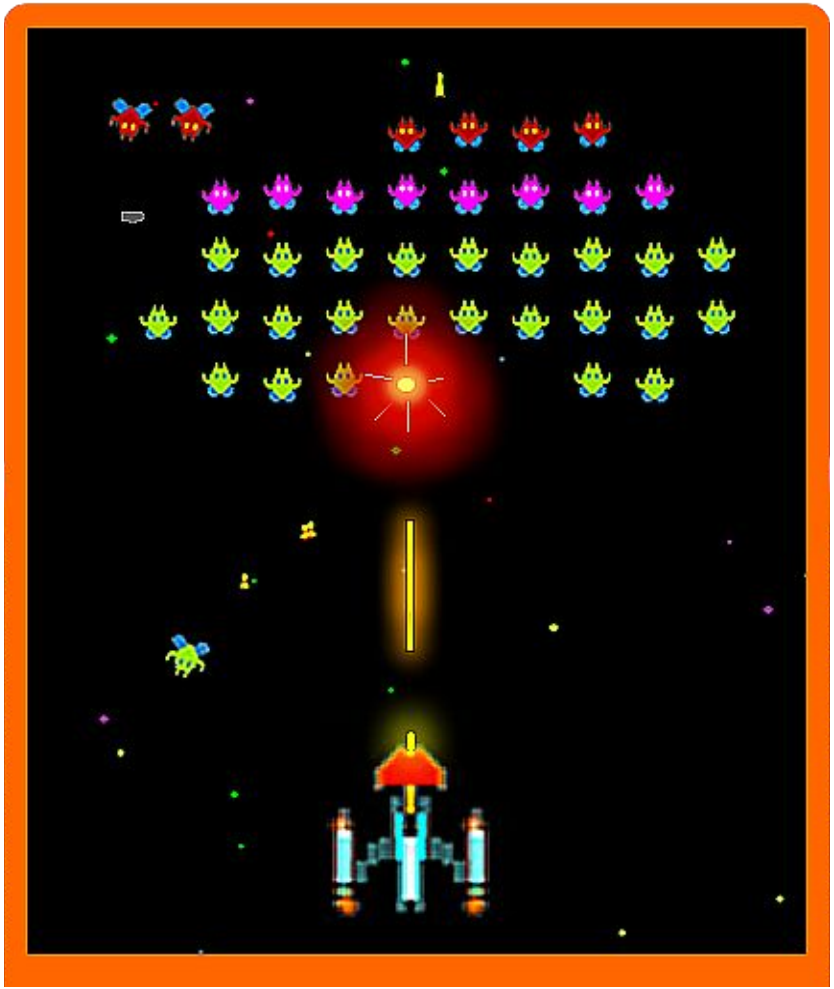
Enemy ships are controlled by the computer.





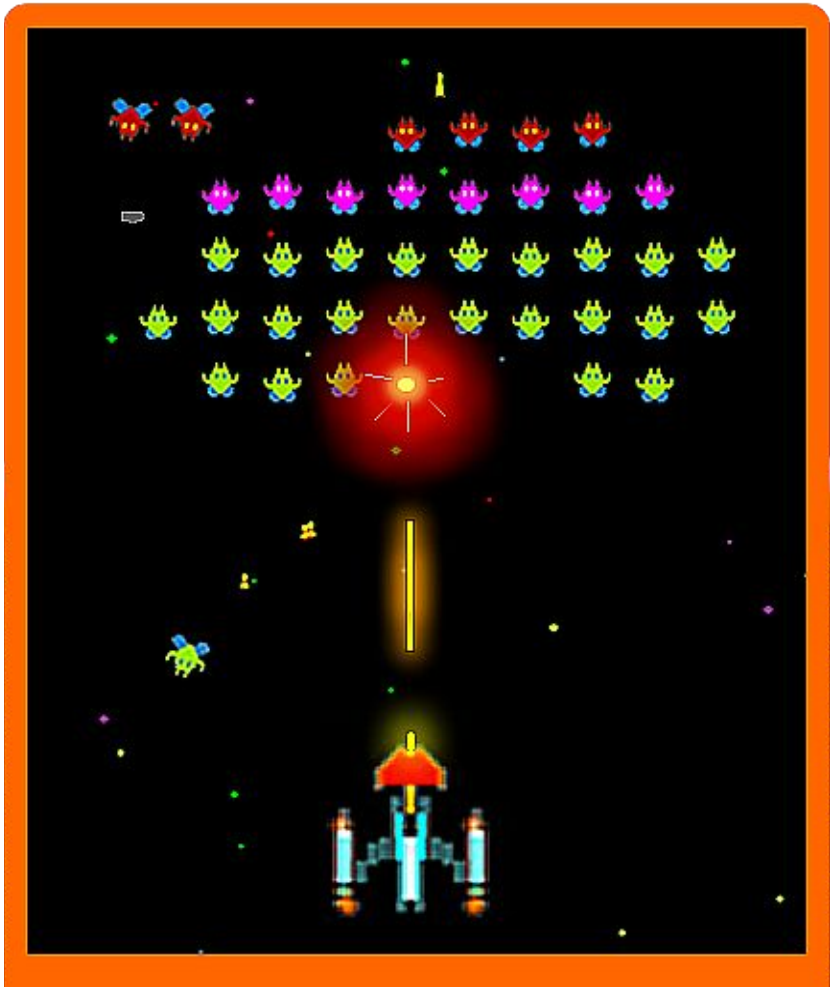
Objects: Pallets

After some time "Power Pallets" or "Energizers" can appear in the game.



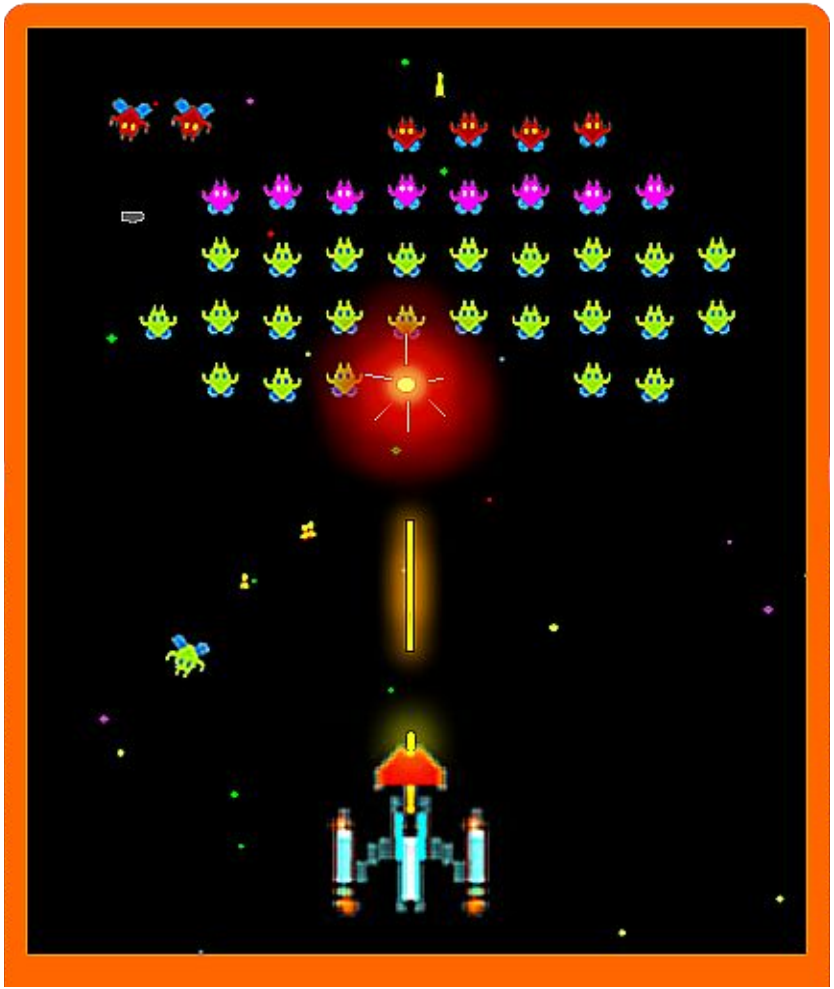
Objects: Walls

Outline represents the end of the maze.



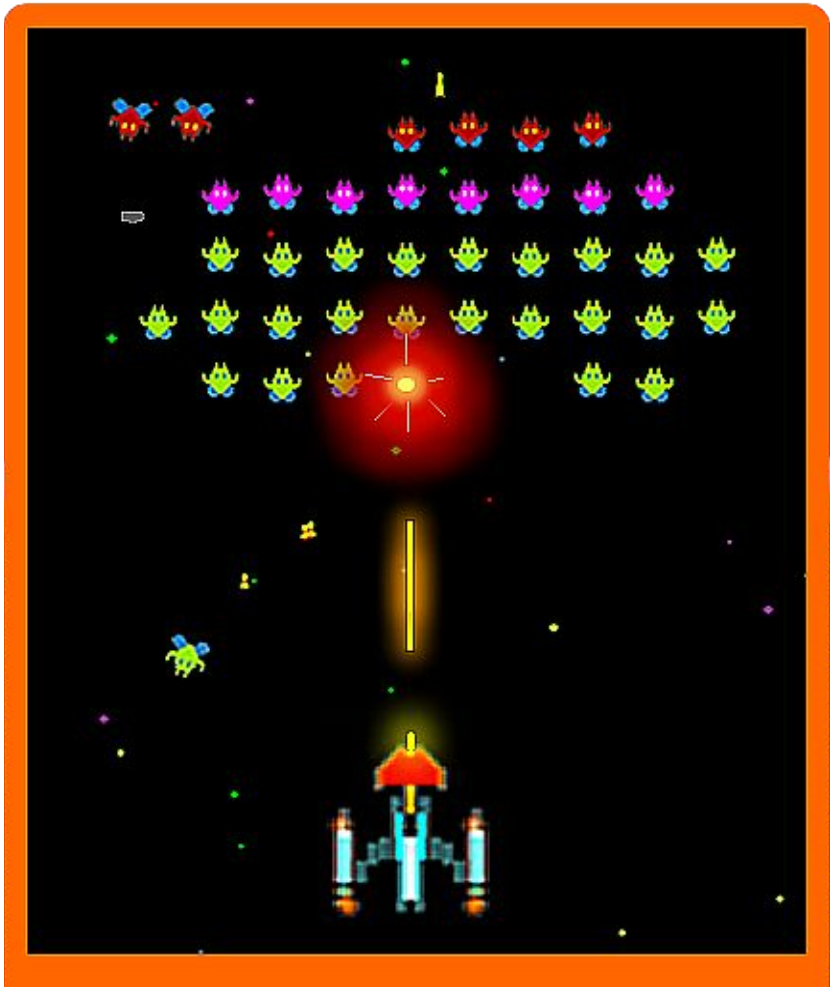
Rules & Interactions

Space Shooter can fire and it has to destroy all enemy ships.
Space Shooter loses a life if he collides with any of the enemies or its fire.
If Space Shooter collides with Power Pallets then the Space Shooter score increases.



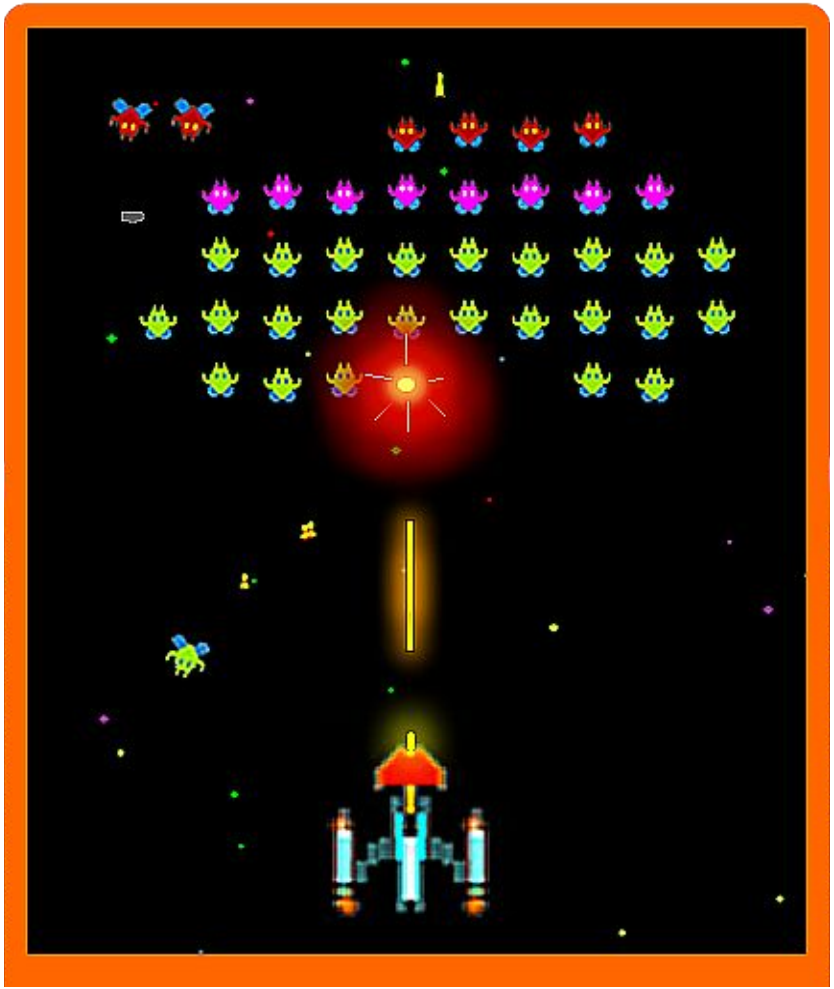
Rules & Interactions

Score also increases when the Space Shooter destroy each enemy.



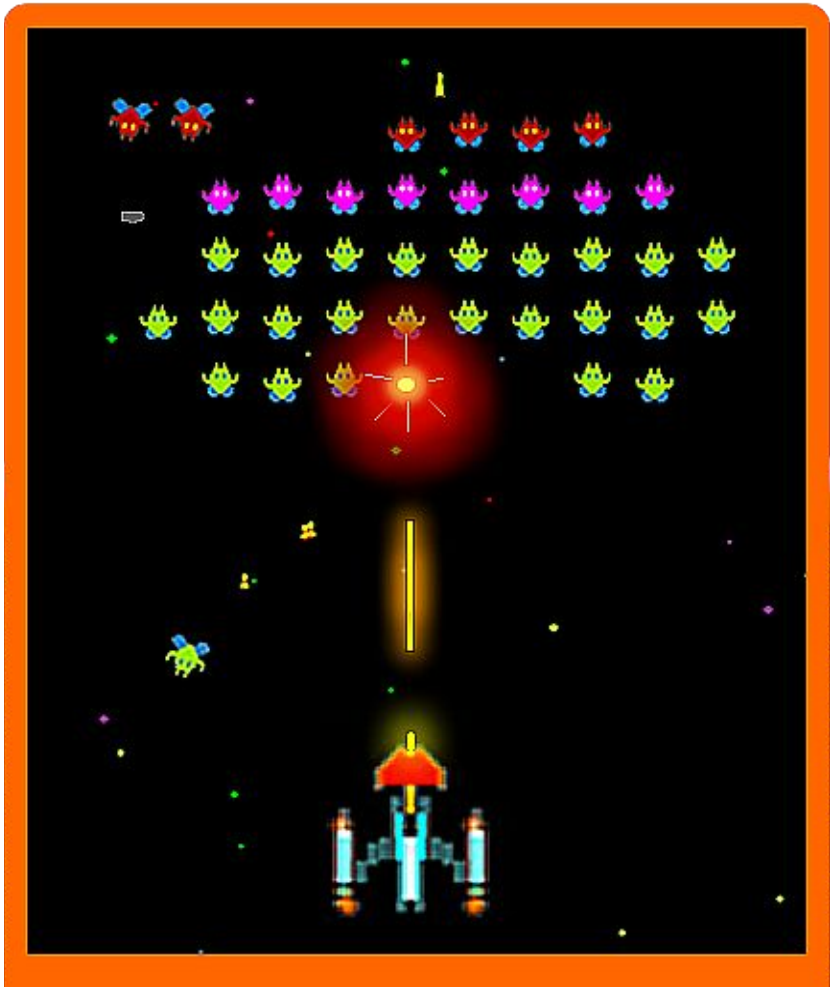
Goal

The goal of the game is to destroy all the enemies that have been put across the maze while avoiding the enemy fires.



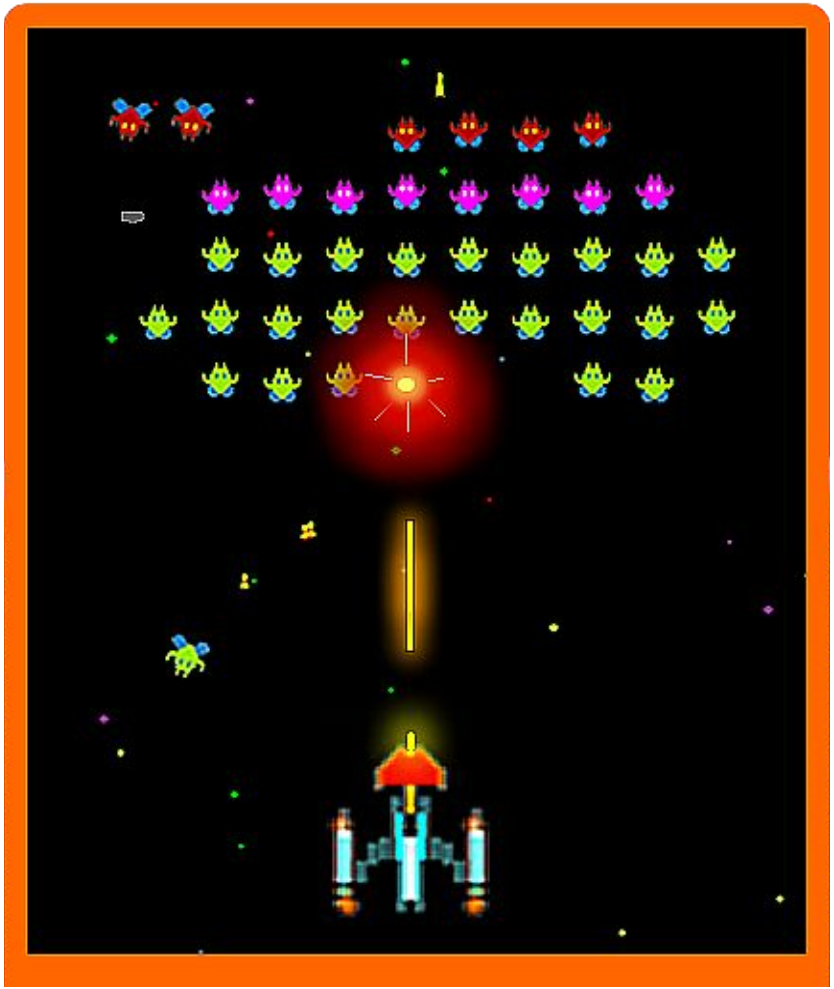
GUI Based Game

This is a GUI-based Game.



GUI Based Game

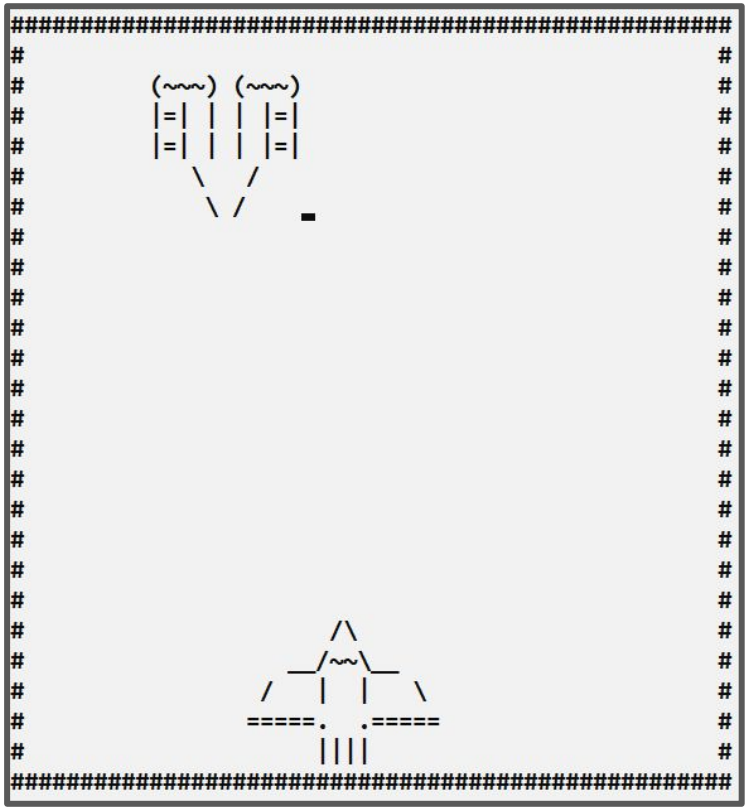
This is a GUI-based Game. We will develop console based game for now.



Console Based Game

The Logic behind both GUI and CLI game is the same which is the most important.

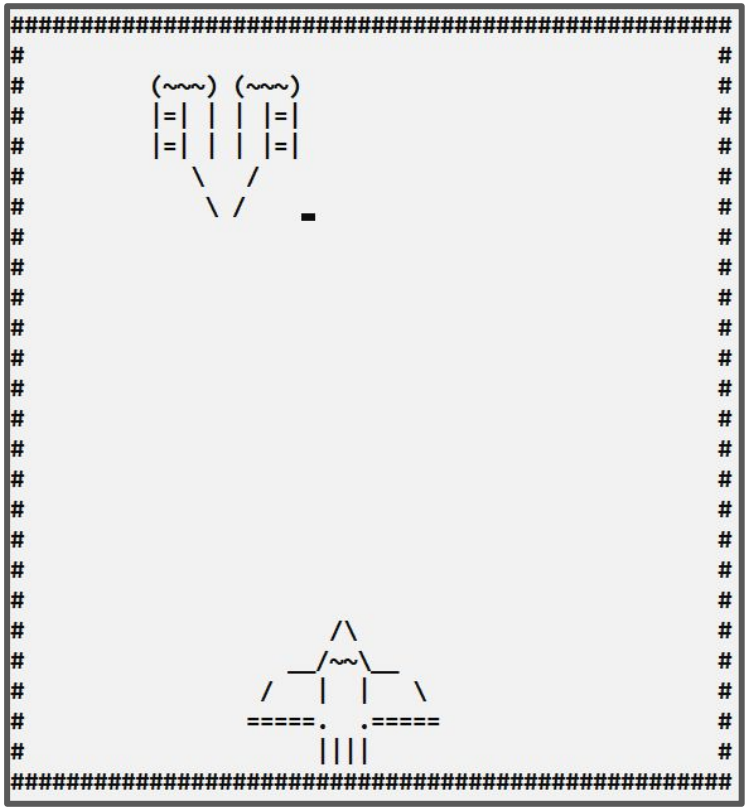
We have already printed the maze on the Console.



Moving Space Shooter

We have moved the Player in the maze horizontally as well as vertically.

But that was for a single Character.
Now, let's see for the complex character.



Moving Space Shooter

Here we are making global variables for enemy's X and Y coordinates as these coordinates have to be used and updated by different functions.

```
#include<iostream>
#include<windows.h>
using namespace std;
void gotoxy(int x, int y);
void printEnemy();
void eraseEnemy();
void moveEnemy();
void printMaze();

int eX = 2, eY = 2;

main()
{
    system("cls");
    printMaze();
    printEnemy();
    while(true)
    {
        moveEnemy();
        Sleep(200);
    }
}
```

Moving Space Shooter

Since the enemy is a complex shape therefore, we will make separate print and erase function for it.

Moving Space Shooter

Since the enemy is a complex shape therefore, we will make separate print and erase function for it.

```
void printEnemy()
{
    gotoxy(eX,eY);
    cout << "(~~~) (~~~)";
    gotoxy(eX,eY+1);
    cout << "|=| | | |=|";
    gotoxy(eX,eY+2);
    cout << "|=| | | |=|";
    gotoxy(eX,eY+3);
    cout << "   \\   /   ";
    gotoxy(eX,eY+4);
    cout << "    \\ /    ";
}
```

```
void eraseEnemy()
{
    gotoxy(eX,eY);
    cout << "           ";
    gotoxy(eX,eY+1);
    cout << "           ";
    gotoxy(eX,eY+2);
    cout << "           ";
    gotoxy(eX,eY+3);
    cout << "           ";
    gotoxy(eX,eY+4);
    cout << "           ";
}
```

Moving Space Shooter

We are using gotoxy function before printing each line therefore it prints on the specific location on the console.

```
void printEnemy()
{
    gotoxy(eX,eY);
    cout << "(~~~) (~~~)";
    gotoxy(eX,eY+1);
    cout << "|=| | | |=|";
    gotoxy(eX,eY+2);
    cout << "|=| | | |=|";
    gotoxy(eX,eY+3);
    cout << "   \\   /   ";
    gotoxy(eX,eY+4);
    cout << "    \\ /    ";
}
```

```
void eraseEnemy()
{
    gotoxy(eX,eY);
    cout << "           ";
    gotoxy(eX,eY+1);
    cout << "           ";
    gotoxy(eX,eY+2);
    cout << "           ";
    gotoxy(eX,eY+3);
    cout << "           ";
    gotoxy(eX,eY+4);
    cout << "           ";
}
```

Moving Space Shooter

Now, the move function calls the printEnemy and eraseEnemy functions.

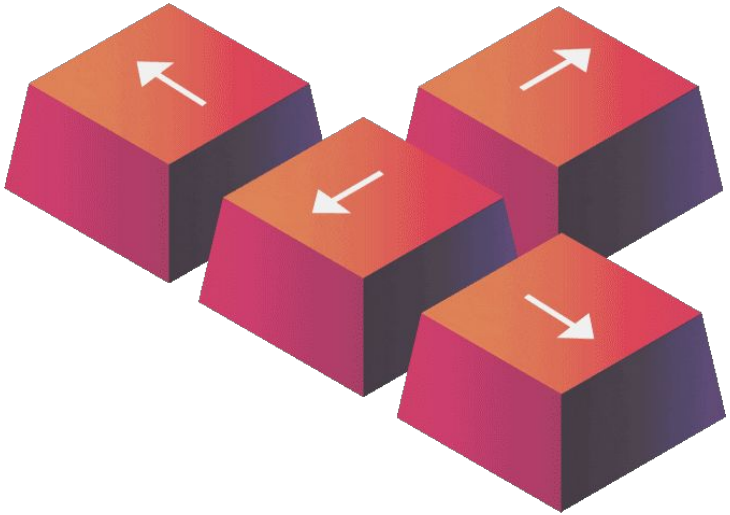
```
void printEnemy()
{
     gotoxy(eX,eY);
     cout << "(~~~) (~~~)";
     gotoxy(eX,eY+1);
     cout << "|=| | | |=|";
     gotoxy(eX,eY+2);
     cout << "|=| | | |=|";
     gotoxy(eX,eY+3);
     cout << "   \\   /   ";
     gotoxy(eX,eY+4);
     cout << "    \\ /    ";
}
```

```
void eraseEnemy()
{
     gotoxy(eX,eY);
     cout << "           ";
     gotoxy(eX,eY+1);
     cout << "           ";
     gotoxy(eX,eY+2);
     cout << "           ";
     gotoxy(eX,eY+3);
     cout << "           ";
     gotoxy(eX,eY+4);
     cout << "           ";
}
```

```
void moveEnemy()
{

     eraseEnemy();
     eX = eX + 1;
     if(eX == 30)
     {
          eX = 2;
     }
     printEnemy();

}
```

Moving Space Shooter using Arrow Keys

Most important thing is to make Space Shooter move with the help of arrow keys.



Moving Space Shooter

Most important thing is to make Space Shooter move with the help of arrow keys.

Player can press:

- Left arrow key (Move Left)
- Right arrow key (Move Right)

Moving Space Shooter

Most important thing is to make Space Shooter move with the help of arrow keys.

Player can press:

- Left arrow key (Move Left)
- Right arrow key (Move Right)

Keys	Movement according to Coordinates
Left	y remains same, x decrements by 1
Right	y remains same, x increments by 1

Space Shooter: Detect Arrow Key

Before changing the location of the Space Shooter we have to detect which arrow key is pressed

Space Shooter: Detect Arrow Key

C++ provides us with a function named GetAsyncKeyState().

GetAsyncKeyState stands for Get Asynchronous Key State.

This function gives information about the key, whether the key was pressed or not at the time when the function was called.

Space Shooter: Detect Arrow Key

But we have to only detect arrow keys.

Space Shooter: Virtual Key Codes

C++ also provides Virtual-key code constants that are used to find the state of the pressed keys.

Codes	Meaning
VK_LEFT	Left arrow key
VK_RIGHT	Right arrow key

Space Shooter: Detect Left arrow Key

Code to detect if the left arrow key is pressed is given by:

```
1  if (GetAsyncKeyState(VK_LEFT))
2  {
3      // Move the Space Shooter Left
4  }
```

GetAsyncKeyState(VK_LEFT) function returns 0 if the key is not pressed and a non zero value if the key is currently pressed.

Space Shooter: windows.h

The definition of GetAsyncKeyState() function is given in the windows.h header file.

```
1  #include <windows.h>
```

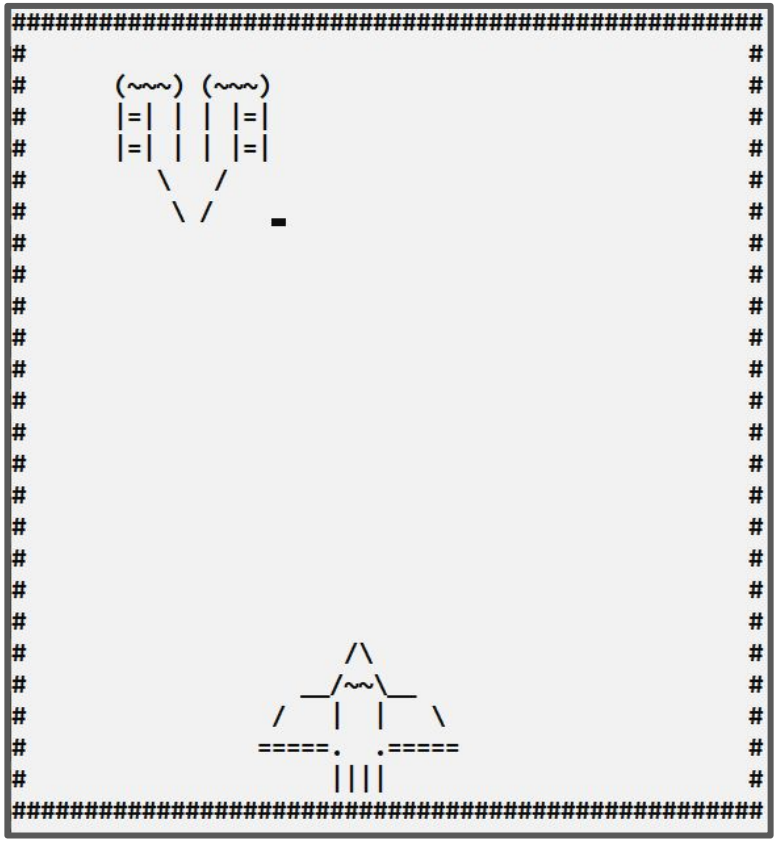
```
#include<iostream>
#include<windows.h>
using namespace std;
void gotoxy(int x, int y);
void printEnemy();
void eraseEnemy();
void erasePlayer();
void printPlayer();
void movePlayerLeft();
void movePlayerRight();
void moveEnemy();
void printMaze();

int eX = 2, eY = 2;
int pX = 20, pY = 20;
```

```
main()
{
    system("cls");
    printMaze();
    printEnemy();
    printPlayer();

    while(true)
    {
        if (GetAsyncKeyState(VK_LEFT))
        {
            movePlayerLeft();
        }
        if (GetAsyncKeyState(VK_RIGHT))
        {
            movePlayerRight();
        }
        moveEnemy();
        Sleep(200);
    }
}
```

```
void erasePlayer()
{
    gotoxy(pX,pY);
    cout << "                    ";
    gotoxy(pX, pY+1);
    cout << "                    ";
    gotoxy(pX, pY+2);
    cout << "                    ";
    gotoxy(pX, pY+3);
    cout << "                    ";
    gotoxy(pX, pY+4);
    cout << "                    ";
}
void printPlayer()
{
    gotoxy(pX,pY);
    cout << "       /\\       ";
    gotoxy(pX, pY+1);
    cout << "    __/~~\\__    ";
    gotoxy(pX, pY+2);
    cout << " /    |  |    \\ ";
    gotoxy(pX, pY+3);
    cout << "=====.  .=====";
    gotoxy(pX, pY+4);
    cout << "      ||||      ";
}
```

```
void movePlayerLeft()
{
    erasePlayer();
    pX = pX - 1;
    printPlayer();
}

void movePlayerRight()
{
    erasePlayer();
    pX = pX + 1;
    printPlayer();
}
```

Space Shooter: Output

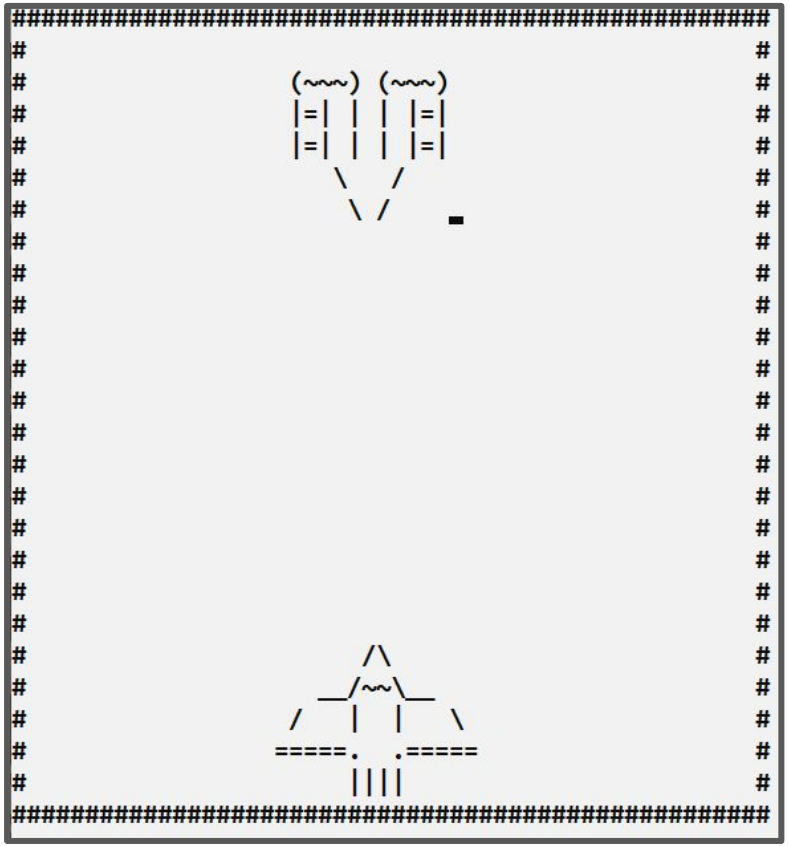


Space Shooter: Activity

Do you see any other problem in this code?

Space Shooter: Collision with Wall

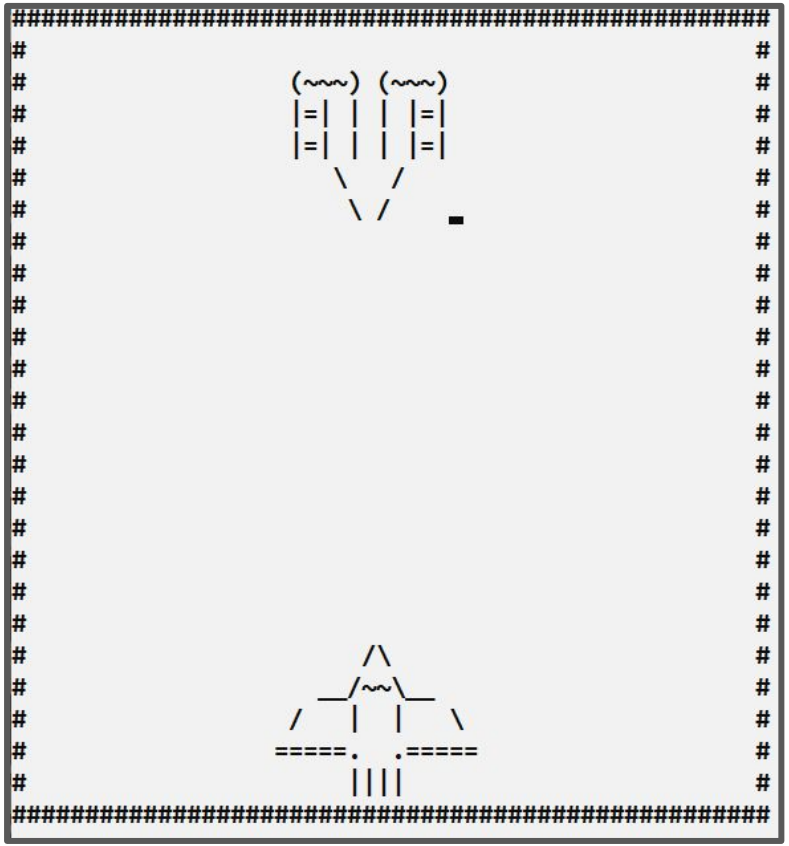
What Happens when Space Shooter reaches any wall?



Space Shooter: Collision with Wall

What Happens when Space Shooter reaches any wall?

are replaced with the Space Shooter Characters.

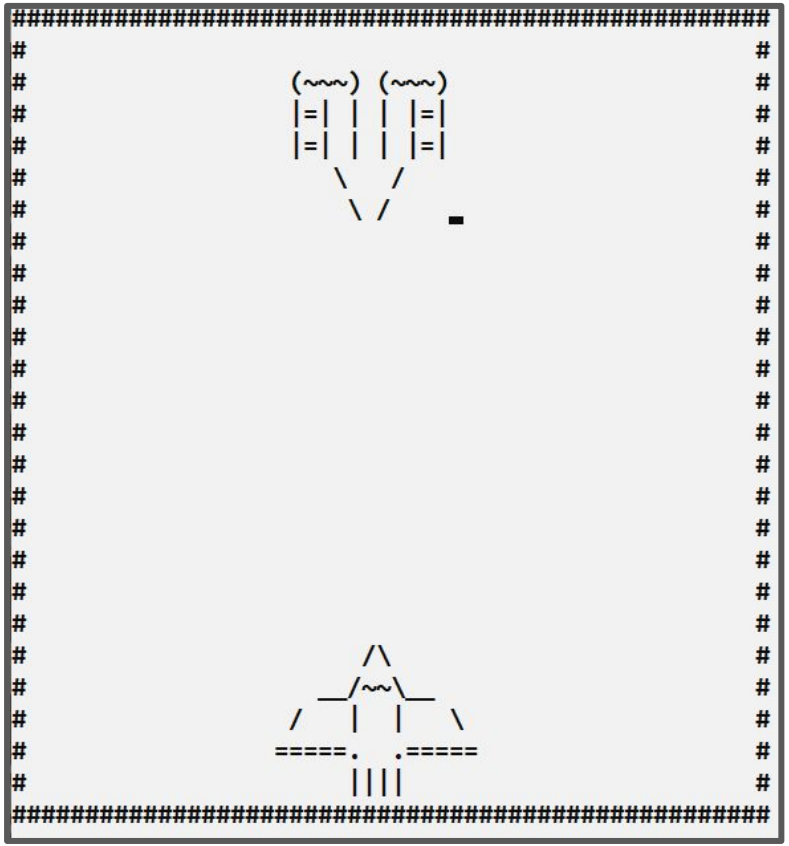


Space Shooter: Collision with Wall

What Happens when Space Shooter reaches any wall?

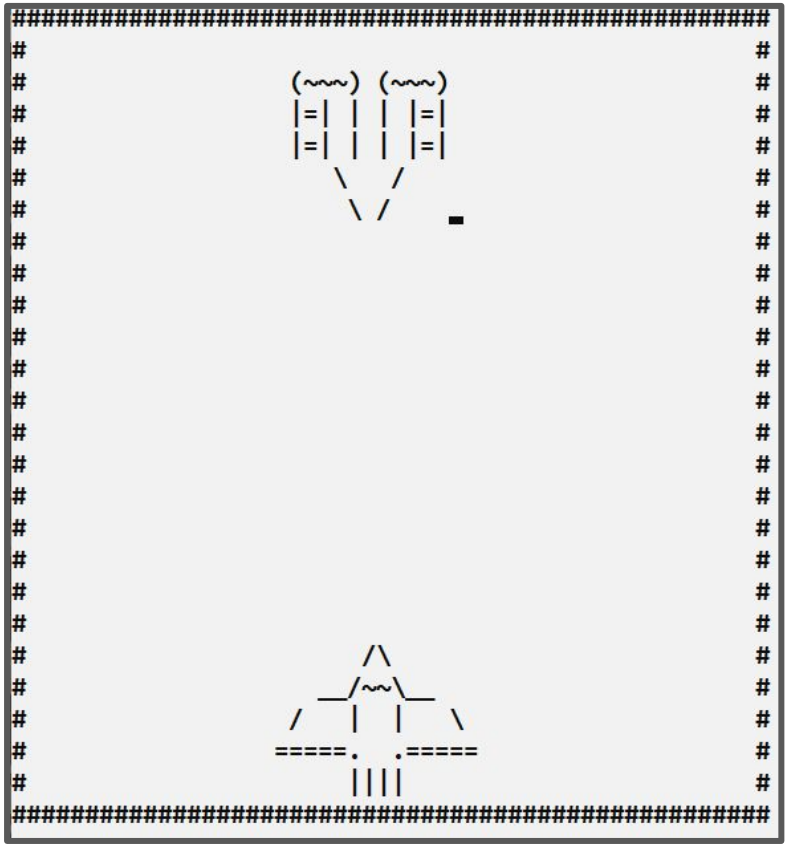
are replaced with the Space Shooter Characters.

What will be the solution?



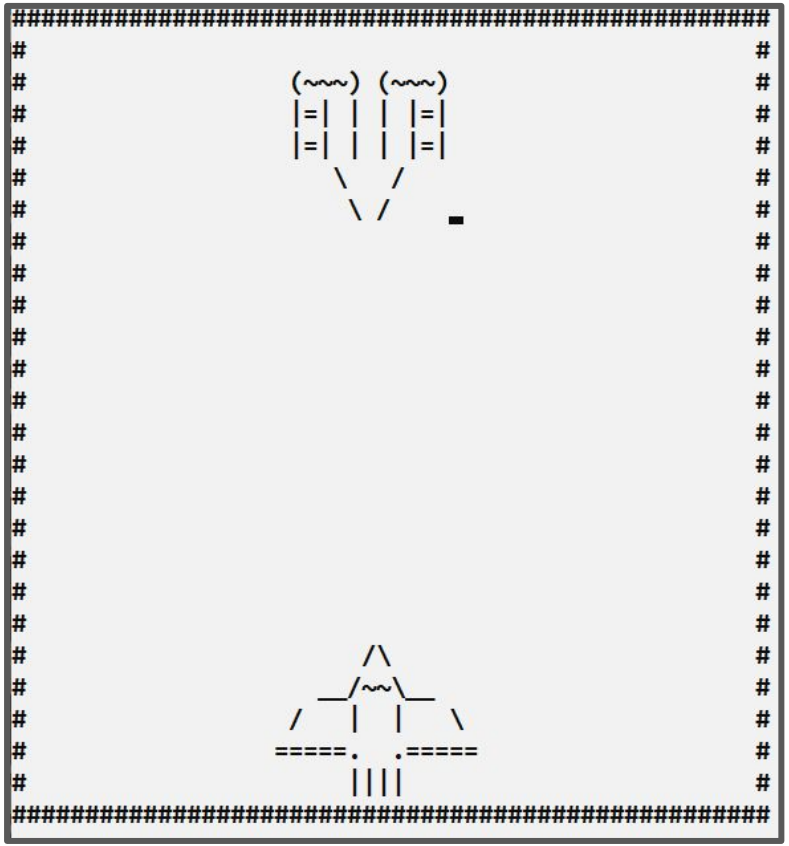
Space Shooter: Collision with Wall

We should only move the Space Ship if the next place from the Left and Right of Space Ship Coordinates contains empty space.



Space Shooter: Collision with Wall

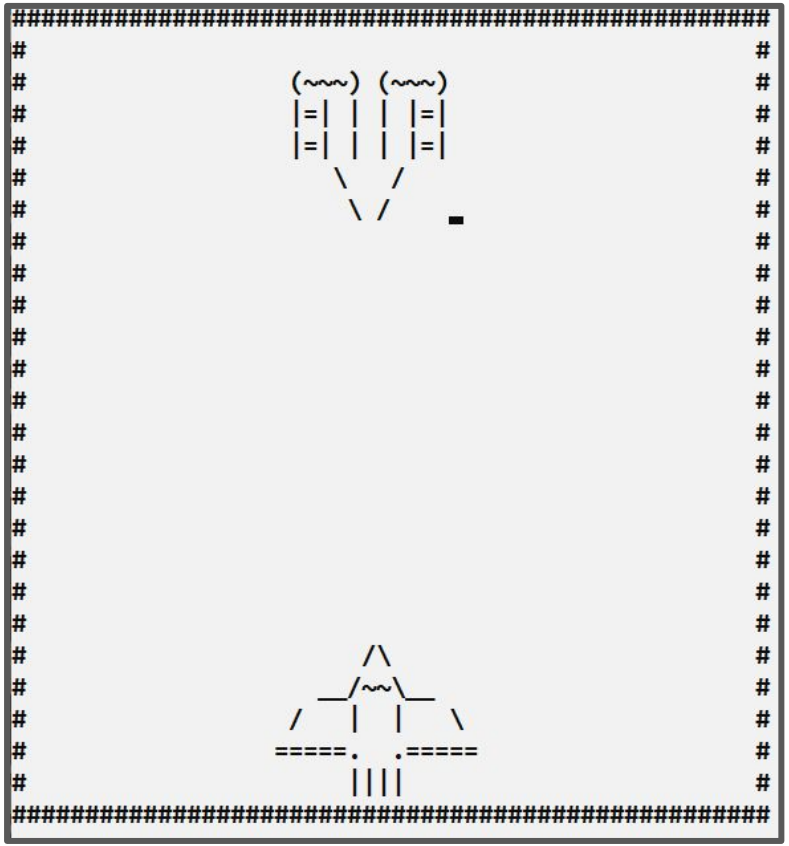
What will be the coordinates that we should check before moving Space Shooter left or right?



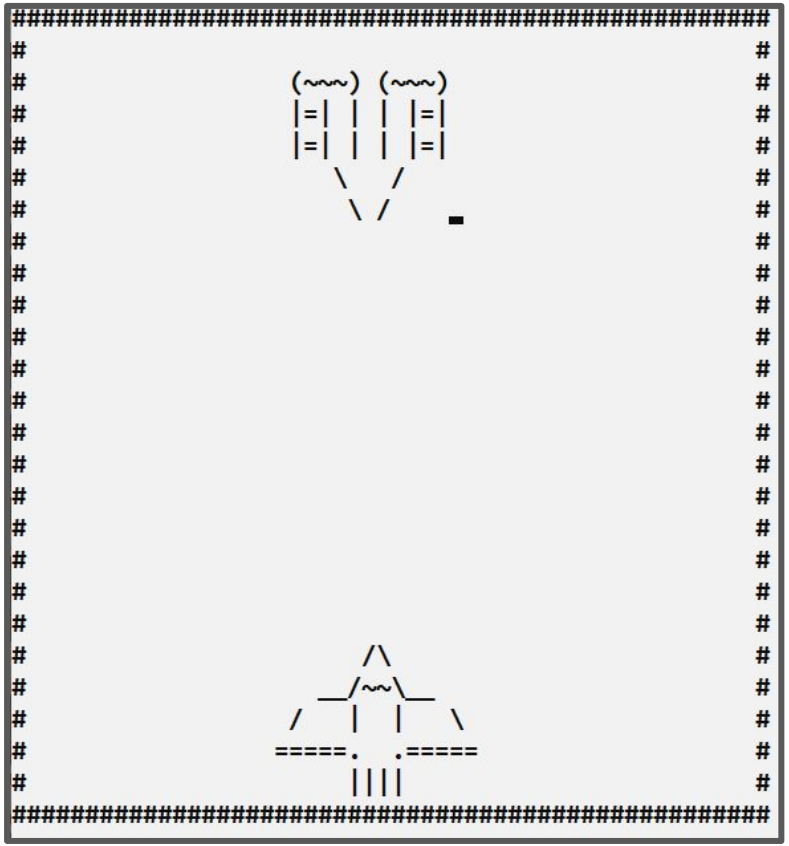
Space Shooter: Collision with Wall

For Left Movement we must check the character at pX - 1.

For Right Movement we must check the character at pX + 15.



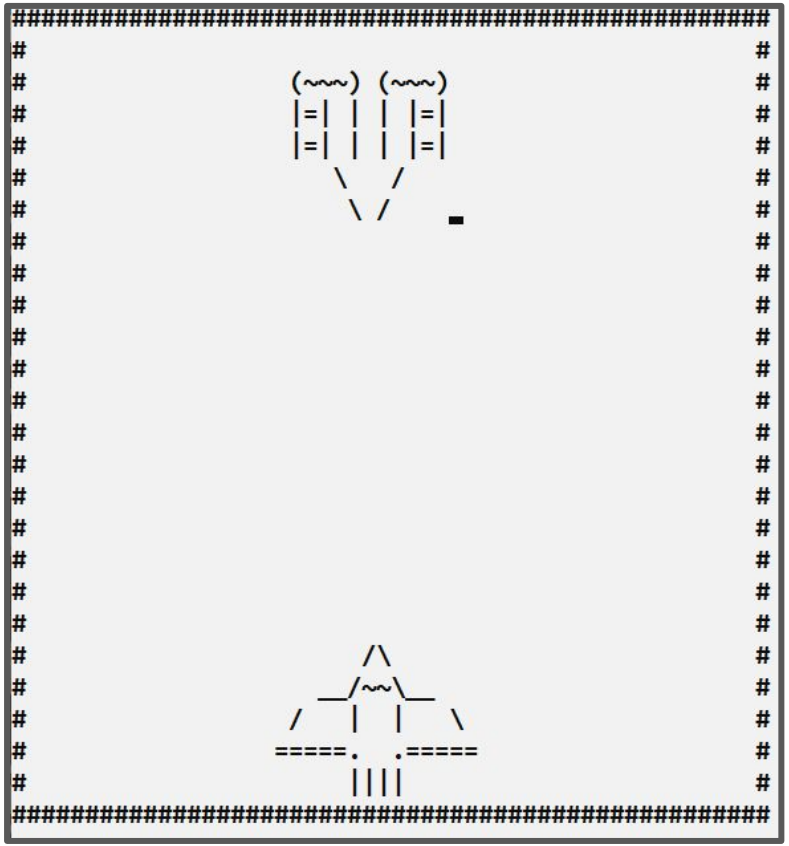
Space Shooter: Collision with Wall



Space Shooter: Collision with Wall

For Right Movement we must check the character at pX + 15.

How to check whether # symbol exist at console position px + 15 ?



Space Shooter: Collision with Wall

In C++, we have a function that will return the character present on console on specific coordinates.

Space Shooter: Collision with Wall

In C++, we have a function that will return the character present on console on specific coordinates.

```
char getCharAtxy(short int x, short int y)
{
    CHAR_INFO ci;
    COORD xy = {0, 0};
    SMALL_RECT rect = {x, y, x, y};
    COORD coordBufSize;
    coordBufSize.X = 1;
    coordBufSize.Y = 1;
    return ReadConsoleOutput(GetStdHandle(STD_OUTPUT_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar
: ' ';
}
```

Space Shooter: Collision with Wall

Again, you do not need to understand internal working of the function. You just have to copy paste in your Code.

```
char getCharAtxy(short int x, short int y)
{
    CHAR_INFO ci;
    COORD xy = {0, 0};
    SMALL_RECT rect = {x, y, x, y};
    COORD coordBufSize;
    coordBufSize.X = 1;
    coordBufSize.Y = 1;
    return ReadConsoleOutput(GetStdHandle(STD_OUTPUT_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar
: ' ';
}
```

Space Shooter: Collision with Wall

Now, our updated Move functionalities will become

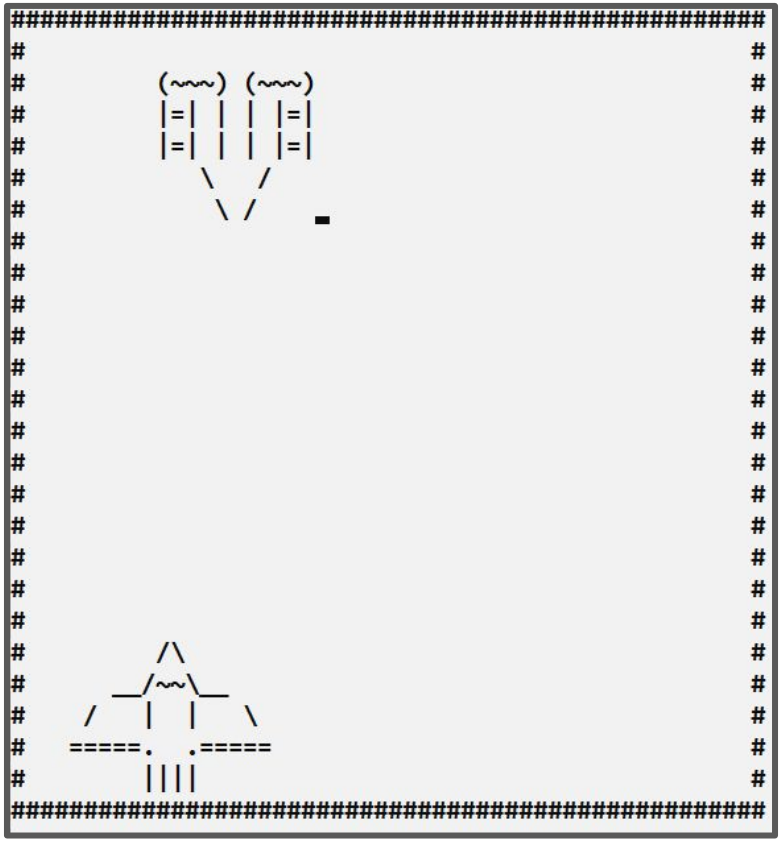
```
void movePlayerRight()
        {
            if (getCharAtxy(pX + 15, pY) == ' ')
            {
                erasePlayer();
                pX = pX + 1;
                printPlayer();
            }
        }
```

Space Shooter: Collision with Wall

Now, our updated Move functionalities will become

```
void movePlayerLeft()
{
            if (getCharAtxy(pX - 1, pY) == ' ')
            {
                erasePlayer();
                pX = pX - 1;
                printPlayer();

            }
}
```

Space Shooter: Output



Learning Objective

Write a C++ program to move a game object on the console using arrow keys and detect collision.

Conclusion

- GetAsyncKeyState stands for Get Asynchronous Key State. This function gives information whether the key was pressed or not at the time when the function was called.
- Virtual key codes for the arrow keys and their meanings are given below.

Codes	Meaning
VK_LEFT	Left arrow key
VK_RIGHT	Right arrow key
VK_UP	Up Arrow key
VK_DOWN	Down arrow key

Conclusion

- Syntax to use GetAsyncKeyState() function is as follows:

```
#include <windows.h>
main()
{
     if (GetAsyncKeyState(VK_Code))
     {
          // Do something
     }
}
```

Conclusion

- Function to read a character from the console is as follows:

```
char getCharAtxy(short int x, short int y)
{
    CHAR_INFO ci;
    COORD xy = {0, 0};
    SMALL_RECT rect = {x, y, x, y};
    COORD coordBufSize;
    coordBufSize.X = 1;
    coordBufSize.Y = 1;
    return ReadConsoleOutput(GetStdHandle(STD_OUTPUT_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar
: ' ';
}
```

Self Assessment: (Video Profile Activity)

1. Now your task is to make the Space Shooter for a larger grid world. Include the Up and Down movement functionality and if the Space Shooter strucks the wall, it does not change its position.
2. Also, Add 2 Enemies in the Grid and move one of them vertically and other horizontally.
3. Also, add the functionality if Space Shooter collides with Enemy game should be over, and if it collides with pallets then its score increases.
4. Also try to implement the logic of Space Shooter fire if the player presses the Space key.