|  |
| --- |
| SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE |
| Programming Paradigms  Mini Project |
| 'SURVIVAL' |
|  |
| Rishabh Kaushick, 15070121152  Roger Noronha, 15070121153  Shahbegh Singh, 15070121161  Yash Saboo, 15070121170 |
| Date of Submission 10/21/2016 |

|  |
| --- |
|  |

# Abstract

Survival is an 'endless running' C++ game. Players of the game take the role as the driver of the car who runs down the lanes to dodge the obstacles.The objective of the game is to survive the longest and get the maximum score while running through an endless game world. Blocks must be avoided by performing well timed movement of the car (swipe left and right). The game ends when the player either stumbles and is apprehended or crashes directly into an obstacle.

Table of Contents

[Abstract 2](#_Toc464823378)

[Rules 4](#_Toc464823379)

[Header Files, Classes & Functions 4](#_Toc464823380)

[Header File 4](#_Toc464823381)

[Class Game2 5](#_Toc464823382)

[Functions 5](#_Toc464823383)

[void perform() 5](#_Toc464823384)

[void obstacle(int,int) 5](#_Toc464823385)

[void car(int,int) 5](#_Toc464823386)

[void obstacleCreator() 5](#_Toc464823387)

[int gameover(int) 6](#_Toc464823388)

[int rand\_int(int a,int b) 6](#_Toc464823389)

[Class Game 1 6](#_Toc464823390)

[Functions 6](#_Toc464823391)

[void firstpage() 6](#_Toc464823392)

[void lines() 7](#_Toc464823393)

[void car(int,int) 7](#_Toc464823394)

[void obstacle1(int) 7](#_Toc464823395)

[void obstacle2(int) 7](#_Toc464823396)

[void hit(int,int,int) 7](#_Toc464823397)

[void perform1() 7](#_Toc464823398)

[void lastpage() 7](#_Toc464823399)

[Structure 7](#_Toc464823400)

[Structure Queue 7](#_Toc464823401)

[Functions for the structure 8](#_Toc464823402)

[OOP Concepts 9](#_Toc464823403)

[Algorithm 10](#_Toc464823404)

[EASY GAME LOGIC 11](#_Toc464823405)

[HARD GAME LOGIC 14](#_Toc464823406)

[SCREEN SHOTS 16](#_Toc464823407)

# Rules

As the name of this game suggests the main aim is to survive. In the main menu there are two difficulty levels-easy / hard. If you choose either one of them you are directed to the game.

So how do we play the game?

1. You'll get a car.
2. You've got to dodge the obstacles that come your way.
3. The more obstacles you dodge the more points you earn.
4. As you progress the game goes faster making obstacles hard to dodge.

You may be thinking, 'How many lives do you get?'

JUST ONE.

Why?

Because its SURIVIAL!

The whole point of the game is to survive as long as you can to break the high score. If you crash you're out of the game.

GOOD LUCK, AND MAY THE ODDS BE EVER IN YOUR FAVOR

# Header Files, Classes & Functions

## Header File

<stdio.h> - sprintf

<stdlib.h> - exit, srand

<graphics.h> - initgraph,settextstyle,

outtextxy,rectangle,arc,  
 getmaxx,getmaxy,setbkcolor,  
 cleardevice,line

<dos.h> - delay

<time.h> - time

<conio.h> - getch

<iostream.h> - cout

## Class Game2

This class contains following functions:

void perform()  
void obstacle(int,int)  
void car(int,int)  
void obstacleCreator()  
int gameover(int)

## Functions

### void perform()

It performs all the game algorithmic task cumulatively.

Additionally, it also performs some tasks exclusively like:

Sets background colour

Measures the distance covered

Checks if the obstacle has past its life span

Calculate and prints scores

### void obstacle(int,int)

A lane parameter is passed to display the obstacle in particular lane.

A ‘i’ parameter is passed which acts as the amount of distance, the obstacle covered, with respect to Y-Axis.

### void car(int,int)

It prints the car with its respective lane.

### void obstacleCreator()

It is a function which creates obstacles randomly with respect to the difficulty.

Randomly refers that randomly in any lane.

Difficulty refers to the number of obstacles created with respect to the amount of distance covered shown on the table on the next page.

|  |  |
| --- | --- |
| **Probability of Creation of Obstacle** | **Distance** |
| 3/8 | Distance<300 |
| 3/6 | 300<Distance<600 |
| 3/5 | 600<Distance<900 |
| 3/4 | Distance>900 |

### int gameover(int)

It checks if the car hit the obstacle by comparing both of their positions.

### int rand\_int(int a,int b)

It calculates the random integer, ranging between a and b.

## Class Game 1

This class consists of the following functions-

void perform1();

void lines();

void obstacle1(int);

void obstacle2(int);

void car1(int,int);

void hit(int,int,int);

void firstpage();

void lastpage();

## Functions

### void firstpage()

It basically shows the menu where the user decides between the easy and difficult game mode.

Based on user’s choice(i.e. <- left or -> right) respective game mode will take place.

### void lines()

This function shows 4 lines thereby setting the three lanes in which the car can switchline(int,int,int,int) function used which has four parameters or 2 coordinates i.e. initial and final co-ordinate of line.

### void car(int,int)

This function will display car in its respective lane.

### void obstacle1(int)

Generates rectangle type obstacle using the rectangle function and the parameter passed tells which lane to display the obstacle in.

Specifically generates one obstacle in one lane at a time.

### void obstacle2(int)

Follows the same algorithm as obstacle1(int) but two obstacles are generates in two different lanes at a time

### void hit(int,int,int)

It will detect when obstacle and car collide with each by checking the coordinates of car and obstacle.

First two parameters are that of the obstacles and third is of the car.

### void perform1()

This function is the heart of the program as the main algorithm is being followed in this function so as to how the game is being played and nearly all other functions are getting invoked in this function.

### void lastpage()

Here, the Game Over screen appears after the car and obstacle have bumped.

Also , the final score is also shown in this function.

## Structure

### Structure Queue

This structure is used to keep a track of the obstacles.

It is a normal linear queue having two array variables which stores lane of each obstacle and position of each obstacle it is at.

It has following variable:

int l[MAX];

int top;

int i[MAX];

int bottom;

It has following functions:

void initialise()

int empty()

int full()

void enqueue(int n)

void dequeue()

void display()

### Functions for the structure

#### int empty()

* It checks if the queue is empty or not.

#### int full()

* It checks if the queue is full or not.

#### void enqueue(int n)

* It checks if the lane passed as an argument is among the values 1, 2 and 3.
* It checks two things before enqueuing an obstacle
  + If the lane it is to inserted in has no obstacle present between 0 and 150 with respect to Y-axis.
  + If two obstacles are already present in other lane lanes between 0 and 90 with respect to Y-axix.

#### void dequeue()

* It dequeues the obstacle if the life span of the obstacle has been completed.

#### void display()

* It displays the values stored in the queue.

# OOP Concepts

## Class and objects

Class game1 and game are the two classes and game has object g through which functions are called in main

## Inheritance

Class game inherits class game1 publically and is able to access functions such as firstpage(),lastpage() and lines() thereby providing reusability of functions.

# Algorithm

## EASY GAME LOGIC

False

## HARD GAME LOGIC

# SCREEN SHOTS









