Table of Contents

1. Gaming in the Field of Software Engineering

In the fast growing field of software engineering and development and even more rapidly growing sector of game development the future is hard to predict. In general software project is a project focusing on the creation of software. Consequently, Success can be measured by taking a look at the resulting software. In a game project, the product is a game. But and here comes the point: A game is much more than just its software. It has to provide content to become enjoyable. Just like a web server: without content the server is useless, and the quality cannot be measured. This has an important effect on the game project as a whole. The software part of the project is not the only one, and it must be considered in connection to all other parts: The environment of the game, the story, characters, game plays, the artwork, and so on.

2. Proposed gaming system

The Tic Tac Toe game is a game for two players, called "X" and "O", who take turns marking the spaces in a 3×3 grid. The player who succeeded in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. The Tic Tac Toe is a great way to pass your free time whether you're standing in a line or spending time with your kids. Stop wasting paper and save trees. Because of the simplicity of Tic Tac Toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence.

3. Why This Project?

Since the 1970s [10], people started to take interest in using their computers as an entertainment environment, thus, the multi billion game industry [3] was starting to take shape. Having presented earlier the sum of money this industry produces, I decided to have a go and create a game of my own. As a kid, I was always fascinated by the idea of becoming a game developer, but, as years went by, I have realized this is not exactly what programming and computer science, as a practice, are about and I dropped the idea. However, the third year project offered me the possibility to try and achieve one of my childhood’s dreams and I couldn’t resist the temptation.

4. Objectives of the Gaming System

The game is developed for full-time entertainment and enthusiasms. It teaches the Gamer to be alert at every situation he/she faces, because if the Gamer is not fully alert and notice the saucer fire he/she must be hit by the saucer-bombs.

Though the proposed game is an action game, it doesn't involve direct violence. No zombie killing, animal killing, or human killing is performed in the game. So it can also be viewed as a non-violence game.

Kids can also play this game, because the design of the game is very simple, controlling the game is very easy – pressing some neighboring keys of the keyboard.

5. Scope of Our Game

This Report describes all the requirements for the project. The purpose of this research is to provide a virtual image for the combination of both structured and unstructured information of my project "Tic Tac Toe". This is a single-player strategy game on the Windows platform. The player will progress through levels which require precise manipulation of the environment, though the game Encourages creativity and daring via branching pathways. The episodic structure of the game facilitates the pace of the story. I demonstrate the action flow between inputs, script, display (output).

6. Features

(a) Single Player

(b) Multi-Player

(c) Background Music

(d) Different themes

7. Planning and Scheduling

Project planning and scheduling is a part of project management. The project planning stage requires several inputs, including conceptual proposals, project schedules. The development of this project is not successfully done without proper planning and scheduling. Project planning and scheduling is very important stage for us.

(a) Analysis: The maximum time for analysis phase of this project is 2 days.

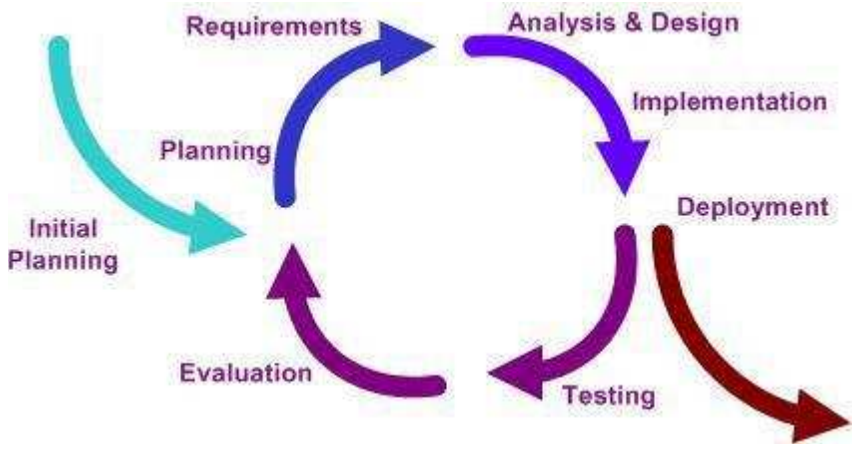
(b) Design: The maximum time for design phase of this project is 5 days.

(c) Implementation: The maximum time for implementation phase of this project is 7 days.

(d) Testing: The maximum time for testing phase of this project is 1 days.

8. Development Process

We planned the project over a period of 15 days and divided it into four iterations. We planned the first iteration for analysis, second iteration for game design, third iteration for coding and the final iteration for the product.



In the first iteration, we focused on Project analysis determined as the first planned milestone of the project. Analysis is essential for starting of upcoming milestones and delivering a finished project on time. Successful completion of a project is heavily dependent on effective analysis.

The second iteration started by brainstorming among group members on what the game would be. Each group member denoted the attributes or properties of the game that one dreamed to implement. We gathered suggestions together and chose the ones that was possible to be implemented within a 15 days project time. As soon as the game concept became clear, we made some early decisions on basic requirements of the project in order to more easily reach the development goals. Game design document was meant to be a living document. In other words, throughout the production process the document was updated, if needed.

In the third iteration coding in Java using Android studio was under way. Therefore, most of time in this iteration was dedicated for internal training sessions. In this iteration, we needed to achieve four milestones each was dependent on the previous one.

Last iteration was planned for testing and finalizing the product. The testing process is an iterative process. We performed the testing process in four iterations. The successful testing process of software requires a good plan. Therefore, after the requirements of the project are confirmed, the future testing of the system and the code were planned. The test plan provided information on how and when the testing will be executed. In the second iteration, test cases were designed for the planned tests. In iteration three, the designed test cases were executed alongside the module testing and usability testing. During the last iteration, according to the result of the tests, the test reports were documented properly and the bugs were reported after the testing is completed.

9. Risk Management

A common definition of risk is an uncertain event that if it occurs, can have a positive or negative effect on a project's goals. It helps us to achieve the project's objectives, thus ensuring the successful completion of the project. For successful development of this project we have to need to identify the possible risk. The possible risk for this project is described in below:

(a) The probability of moving away our-self from this project before it is finish is low.

(b) The probability of user acceptance is great.

(c) The probability of requirement can't come in the time is low.

(d) The probability of marketing the product system is great.

(e) The probability of technology components aren't fit for purpose of this project is low.

(f) The probability of selecting low quality requirements are low.

(g) The probability of take wrong decisions are low.

(h) The probability of does not complete this project within a limited time is low.

10. System Specifications

Most of the computer games require high configurations of computer. But in the case of the proposed gaming system, the system requirements is not that much.

The minimum systems requirements for the proposed project "Tic Tac Toe" game is mentioned following.

(a) Operating System: Android 4.0

(b) Processor: 1.2 GHz

(c) RAM: 512MB

(d) Storage: 100 MB

11. Development Tools and Programming Languages

(a) Android Studio

(b) Java Development Kit (JDK)

(c) Java

12. Expected Requirements

These requirements are implicit to the system and may be so fundamental that the actor/gamer/ relevant people does not explicitly state them .Their absence will be a cause for dissatisfaction.

1. Develop system within limited cost.

2. Maximum high definition.

3. Minimum hardware requirements which is relevant for this game.

4. Design whole system with efficient manner.

13. Development Tasks

1. Android Studio will bring all of the following codes together to create the game. It will also handle AI and physics routines.

2. Graphics engine will be responsible for rendering text, 2D images, and 3D models on screen.

(a) Drawing models

(b) Drawing sprites

(c) Drawing text

(d) Texturing models

(e) Animation

3. Sound engine will be responsible for playing music and sound effects.

(a) Multithreading

(b) Playing sounds

4. Input engine will be responsible for transferring mouse and keyboard input upon request to the game engine.

(a) Retrieving Input

5. Menu Engine will handle all menus in game.

14. System Design MVC Model

MVC model is one of the software architecture in the software engineering. Basically it is consisted of 3 parts.

Model: package with the application's logic related to data as well as data processing method, which directly manipulate data as well as record the action to be performed with the method of implementation.

View: the designed display. In general, it does not have any logic programming. But it will refresh the display constantly to show any message or annotation to the user. It also displays the result given by the model through Controller.

Controller: play a role in the organization between the different levels is used to control the flow of the application. Controller is responsible for controls entire program logic, manages the relationship of objects and handles the event and responds.

15. Testing

Testing is a process of executing a program with the intent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding. System Testing is an important phase. Testing represents an interesting anomaly for the software. A good test case is one that has a high probability of finding an as undiscovered error.

16. Test Levels

The test approach is divided into three main phases: Module testing, integration testing and system testing. In addition, the system testing includes two sub-phases: functional and usability testing. These planned tests are explained briefly below.

(a) Module testing will perform during coding by using debug messages to check that the written code produces wanted results. An important requirement is that the code will compile with zero bugs.

(b) Integration testing will perform after finish module testing in order to validate if each module can work fine with each other. Integration Test proves that system works as integrated unit when all the fixes are complete.

(c) System testing includes two phases: functional testing and usability testing. These will perform after the product reaches its final version. During functional test phase, the tester will test if the product meets the game requirements. The tester tests the requirements using the use cases listed below in Test Cases section. The usability test will perform to understand how easy it is to learn to play the game. Any person out of the team members will perform this test by playing the game.

17. Code

(a) MainActivity.java

```
package com.example.tuhin.tictactoe;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

    Button b1,b2,b3,b4,b5,b6,b7,b8,b9;

    int turn;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);


        b1=(Button)findViewById(R.id.b1);

        b2=(Button)findViewById(R.id.b2);

        b3=(Button)findViewById(R.id.b3);
```

```
b4=(Button)findViewById(R.id.b4);
b5=(Button)findViewById(R.id.b5);
b6=(Button)findViewById(R.id.b6);
b7=(Button)findViewById(R.id.b7);
b8=(Button)findViewById(R.id.b8);
b9=(Button)findViewById(R.id.b9);
turn=1;


b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b1.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b1.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b1.setText("O");
            }
        }
        endGame();
```

```
    }
});


b2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b2.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b2.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b2.setText("O");
            }
        }
        endGame();
    }
});
b3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        if(b3.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b3.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b3.setText("O");
            }
        }
        endGame();
    }
});
b4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b4.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b4.setText("X");
            } else if (turn == 2) {
                turn = 1;
```

```
                b4.setText("O");
            }
        }
        endGame();
    }
});
b5.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b5.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b5.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b5.setText("O");
            }
        }
        endGame();
    }
});
```

```
b6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b6.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b6.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b6.setText("O");
            }
        }
        endGame();
    }
});
b7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b7.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
```

```
                b7.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b7.setText("O");
            }
        }
        endGame();
    }
});
b8.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(b8.getText().toString().equals("")) {
            if (turn == 1) {
                turn = 2;
                b8.setText("X");
            } else if (turn == 2) {
                turn = 1;
                b8.setText("O");
            }
        }
```

```
            endGame();
        }
    });
    b9.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(b9.getText().toString().equals("")) {
                if (turn == 1) {
                    turn = 2;
                    b9.setText("X");
                } else if (turn == 2) {
                    turn = 1;
                    b9.setText("O");
                }
            }
            endGame();
        }
    });


}
public void endGame(){
```

```
String a,b,c,d,e,f,g,h,i;
   boolean end=false;
   a=b1.getText().toString();
   b=b2.getText().toString();
   c=b3.getText().toString();

   d=b4.getText().toString();
   e=b5.getText().toString();
   f=b6.getText().toString();

   g=b7.getText().toString();
   h=b8.getText().toString();
   i=b9.getText().toString();

   if(a.equals("X") && b.equals("X") && c.equals("X")){
      Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
      end=true;
   }
   if(a.equals("X") && e.equals("X") && i.equals("X")){
      Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
      end=true;
```

```
}
if(a.equals("X") && d.equals("X") && g.equals("X")){
    Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
    end=true;
}
if(b.equals("X") && e.equals("X") && h.equals("X")){
    Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
    end=true;
}
if(c.equals("X") && f.equals("X") && i.equals("X")){
    Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
    end=true;
}
if(d.equals("X") && e.equals("X") && f.equals("X")){
    Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
    end=true;
}
if(g.equals("X") && h.equals("X") && i.equals("X")){
    Toast.makeText(MainActivity.this,"Winner Player X",Toast.LENGTH_LONG).show();
    end=true;
}
```

```
if(a.equals("O") && b.equals("O") && c.equals("O")){
        Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
        end=true;
    }
    if(a.equals("O") && e.equals("O") && i.equals("O")){
        Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
        end=true;
    }
    if(a.equals("O") && d.equals("O") && g.equals("O")){
        Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
        end=true;
    }
    if(b.equals("O") && e.equals("O") && h.equals("O")){
        Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
        end=true;
    }
    if(c.equals("O") && f.equals("O") && i.equals("O")){
        Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
        end=true;
    }
    if(d.equals("O") && e.equals("O") && f.equals("O")){
```

```
            Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
            end=true;
        }
        if(g.equals("O") && h.equals("O") && i.equals("O")){
            Toast.makeText(MainActivity.this,"Winner Player O",Toast.LENGTH_LONG).show();
            end=true;
        }
        if(end) {
            b1.setEnabled(false);
            b2.setEnabled(false);
            b3.setEnabled(false);
            b4.setEnabled(false);
            b5.setEnabled(false);
            b6.setEnabled(false);
            b7.setEnabled(false);
            b8.setEnabled(false);
            b9.setEnabled(false);
        }
    }
}
```

18. Conclusion

The Tic Tac Toe game is most familiar among all the age groups. Intelligence can be a property of any purpose-driven decision maker. This basic idea has been suggested many times. An algorithm of playing Tic Tac Toe has been presented and tested that works in efficient way. Overall the system works without any bugs.