Slice It !

An Academic Project

Presented to

CSCI 4237: SoftwareDesign: HandheldDevices

Department of Computer Science

George Washington University

In Partial Fulfillment

Of the Requirements of the Degree

Master of Science

By

Mahitha Admala

Semester: Fall 2011

Contents

Introduction and Motivation 3

Problem Statement 3

Requirements Analysis 3

Design Document 4

Implementation details, choice of technologies, etc. 4

Test cases. 5

Project Timeline and Tasks distribution 8

References 8

User Manual 10

Introduction and Motivation

Nowadays there are innumerable games being developed in the market. The complexity of each game seems to be rising as we can see the substantial transformation between 'Contra' and the realistic imagery in FIFA 2011. With improvements in clarity of objects on screen comes the price of hardware requirements. Most of the games today need a lot of disk space and as well as graphics card for it to run at its optimal level. In addition, Gaming environment is no longer limited to personal computers; they are being migrated to cell phones and other handheld devices, like iPad and various other tablets.

This motivated us to choose our project - "A simple game for a windows phone".

Problem Statement

The main aim of project was to make the game user friendly and not have too many keys to remember. The second problem we wanted to address was of disk space requirement and graphics card requirement, the game does not require too much disk space nor does it require a graphics card. The project aims to implement a game, which is simple, easy to understand as well as navigate.

Requirements Analysis

Functional Requirements: The game is titled 'Slice It'. It is a very simple game which involves splitting objects which float across the screen via movement of the mouse/screen. If the user manages to slice the objects before they move out-of-view (off the screen) he/she gets a point and the score will be incremented. If the user misses the objects and they move out-of-view (off the screen) he/she doesn't score any points for it. To puzzle the user, there will be instances when a "bomb" would be moving across – the user should avoid touching these objects else the game will end. The user has 30 seconds to play the game and score, a timer runs on the top-left corner of the screen, which indicates the seconds. The game is exited once the 30 seconds elapse.

Non Functional Requirements:

- The game is simple, easy to understand as well as navigate.
- Clarity of images
- User friendly: User does not have too many keys to remember
- System friendly: A few Mbs of disk space requirement. The game does not require a graphics card also.

Technology Requirement:

- The game will be developed using Microsoft XNA Game Studio 4.0.
- Microsoft Visual Studio 2010 SP1: XNA is a development environment which uses the Dot Net Framework and employs the language of C#.
- Windows Phone SDK v7.1

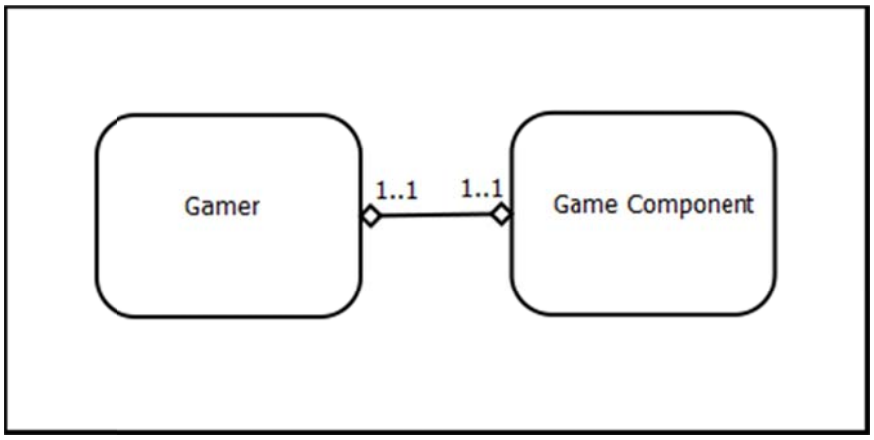
Design Document

High Level Entities:

User/Gamer: The Gamer interacts with the gaming interface via touch/mouse input.

Game Component: This component encapsulates the game functionality along with the user interface for the same. It is responsible for drawing all the game objects/sprites on the screen, updating the game at runtime as well as managing all the functionality. It is also responsible for the movement of the sprites across the screen and also for the sprites to be exterminated from the screen.

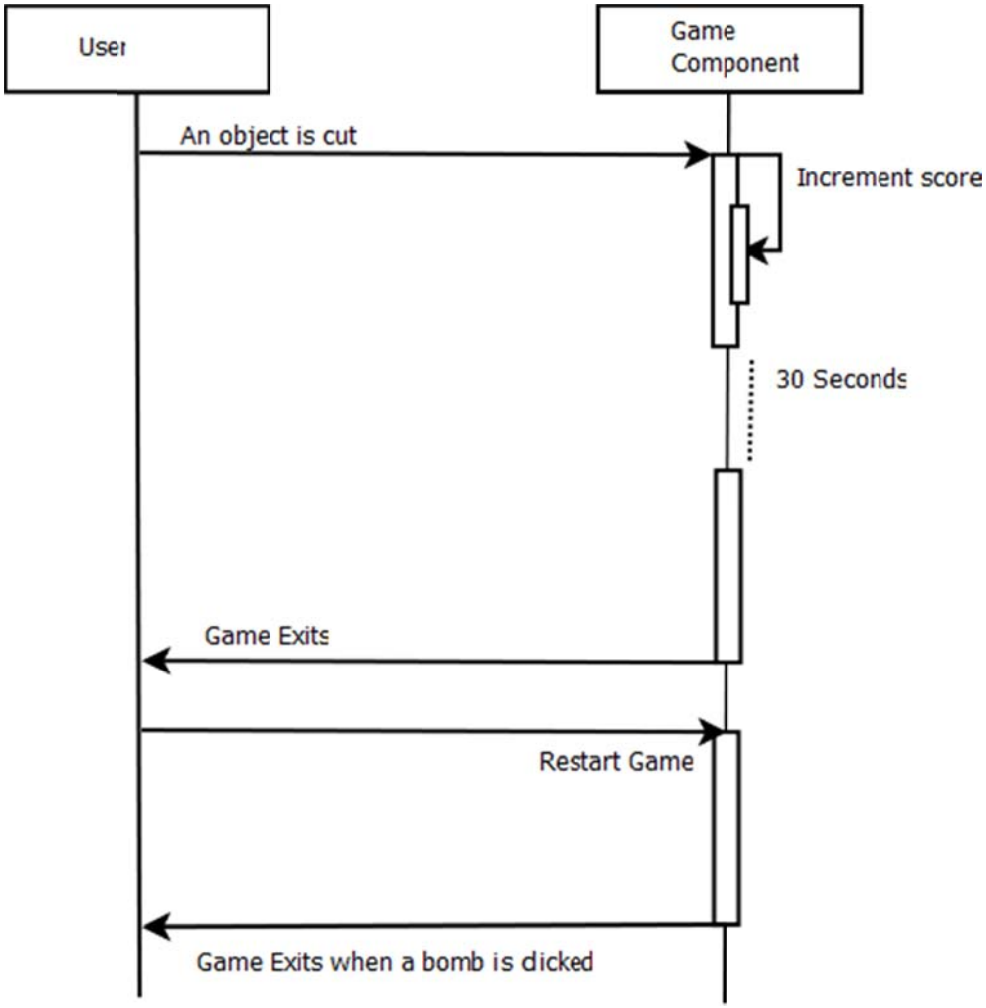
Below diagram provides the high level entities involved in the application and their respective association with each other:



Interaction: The below diagram provides an example of a possible interaction between the user and the game.

1) The Gamer/User is exited from the game, after 30 seconds as it is the time limit for the game.
2) The Score gets incremented for each object the Gamer splits via a mouse click on the respective object.
3) The Gamer/User is exited from the game, if he clicks on a bomb object.

   Below is the Sequence diagram representing probable scenarios, that can occur during game play:



Implementation details, choice of technologies, etc.

The project has been implemented using Microsoft XNA Game Studio 4.0 which works on the Dot Net Framework 4.0 and Microsoft Visual Studio 2010 as the environment.

The structure of the game involves 3 major classes and 1 driving static class. There are 2 arrays of objects of type 'Objectsclass' namely 'firstArray' and 'secondArray'. 'firstArray' defines a set of images which begin moving from the x axis of the screen while 'secondArray' defines images which resume movement from the y axis of the screen. The other 2 array of objects defined are of type 'Cutobjects' namely 'cutfirstArray' and 'cutsecondArray'. The former defines the cut halves of the corresponding whole images defined in 'firstArray' while the latter defines the same for images in 'secondArray'. A whole image represents a complete image while 'cut halves' refers to the 2 images which are the subset of the whole image occurring once the user splits the whole image on the screen. These have been defined below.

1) Program class: This class is the game driver. It initiates the game by creating a local instance of the game and resuming it by invoking its pre-defined function 'Run'.

2) Objectsclass: This class defines whole components of the game i.e. the objects/images before they are cut. It defines the properties of the images:
    a. image: It references the image being displayed
    b. imagePosition: It defines the image position on the display screen
    c. imageVelocity: It controls the speed and direction of image movement on the screen

   The class also defines 2 members of type Boolean ('disappear' and 'isCut') which are used by the 'Game' class in order to determine if the objects have been cut or if the objects have moved out of the display screen dimensions (via the lower horizontal axis). This is used to drive the game on further through time. The class has members ('count', 'score') to keep a track of the number and the type of object being cut which is used to provide the score to the User.

3) Cutobjects: This class represents the sprites/images on being cut. It has the image properties as in the above 'Objectsclass' for each of the 2 halves of the image.
    a. cutImage1 and cutImage2: These 2 members reference the respective image halves predefined once the corresponding whole image is cut.
    b. cutImageposition1 and cutImageposition2: These specify the positions of the cut halves of the whole image
    c. imageVelocity1 and imageVelocity2: The members control the speed and direction of the half image movements on the screen.

   This class also defines a Boolean variable 'disappear' to keep a track if the cut halves move out of the screen dimensions via the lower horizontal axis on the screen.

4) Game1: This class defines the entire game functionality right from displaying the objects on the screen, moving the objects around, as well as splitting the objects. It is the game component which extends the 'Game' class defined in 'Microsoft.XNA.Framework'. It overrides the below 4 functions:
    a. Initialize: This function defines the various non-graphic elements that need to be displayed on the screen before the game resumes its activities. Typical tasks could involve specifying any database related connections. This method is called before the 'Draw' so, the user could experience a delay before the game screen comes up. In this case, the initial position of the 1st 2 objects on the screen is defined.

    b. LoadContent: This method is called once per game run. It is called by the 'Initialize' method just before the 'Draw' method is invoked. As the name suggests, it is used to load any game-specific graphics resource of the game i.e. specify the graphics device or load any objects needed in the game. There is a delay before the initial game screen loads when the code for the method is being executed.
       In this case the images which will be displayed in the game run are passed onto the respective members of the Objectsclass and Cutobjectsclass. Also, the graphics device screen is referenced (spriteBatch).

    c. UnloadContent: This method is used to unload any graphics content that are no longer required.

    d. Draw: This method is used to display the required game components. It is called

anytime a frame needs to be displayed.
In this case the game method is used to the whole images as well the cut halves at various points of the game. There are 2 methods that the Draw method invokes at different points of the game based on a condition:

A. Draw1: This method draws a new whole image based on the selected choice of image by the 'Random Number Generator' based on the below conditions:
    i. A whole image moves out of screen or 2 halves move out of screen
    ii. Draws the whole image on the screen if it hasn't exited the screen dimensions
    iii. Draws the halves if they haven't moved out of the screen.
B. Draw2: This draw function is used to display the cut images at the point the whole image is cut.

e. Update: This method is what controls the game functionality. Whatever the game does have to be defined in this method. Update controls the game state, processes user input as well as updates game data.
In this case, update method controls which of the functions that implement game sprite movement and user input controls are called. It is also responsible for controlling the sprite collision and updating the game timer. The 2 methods that update calls are as below:

A. UpdateImage: This method is used to control the movement of images defined in 'firstArray'. It also controls the user input – i.e. if the user clicks the image via mouse/screen the image is split. This method also keeps track of the score which is incremented by 1 each time the user cuts an object.
B. UpdateSecondArray: This method is used to control the movement of images defined in 'secondArray'. It also controls the user input – i.e. if the user clicks the image via mouse/screen the image is split. This method also keeps track of the score which is incremented by 1 each time the user cuts an object.
C. UpdatecutImage: This method is used to control the movement of half images defined in 'cutfirstArray'.
D. UpdatecutSecondArray: This method is used to control the movement of images defined in 'cutsecondArray'.

The Update method also updates the timer that is displayed on the screen. Once the timer count crosses 30 seconds, the game Exits displaying the score on the screen.

The game exits also in the case when the user cuts a bomb object displaying the score on screen.

Test cases

Please click on the icon below to go to the test case document of the project.

Test Case Document.xlsx

Project Timeline and Tasks distribution

The entire project has been divided into subtasks based on the Gaming functionality and Unit Testing as below:

1) Display/Movement of Single Sprite on Screen
2) Movement of Sprite across the screen boundaries
3) Display of multiple sprites on the screen
4) Mouse Control of Sprites
5) Splitting of Objects
6) Movement of split objects
7) Sprite rebounding
8) Timed Synchronization of Sprites
9) Score Count
10) Time Control of Game
11) Game Exit Control
12) Game integration
13) Unit Testing
14) Documentation

The schedule is displayed below:

S.No	Component	Description	Developer Name	Timeline
1	Display/Movement of Single Sprite on Screen	Brings about the display of a single sprite on the screen and its movement across it	Mahitha	Sept 30 – Oct 5
2	Movement of Sprite across the screen boundaries	Ensures the sprite rebounds across the screen boundaries but exits the screen on reaching the lower level X axis of the screen.	Mahitha	Oct 6 – Oct 11
3	Display of multiple sprites on the screen	This considers displaying multiple sprites on the screen at the simultaneously	Mahitha	Oct 12 – Oct 31
4	Mouse Control of Sprites	Considers the mouse input. It implements the slicing of the objects on encountering the sprites on the screen.	Mahitha	Oct 6 – Oct 13
5	Splitting of Sprites	This involves replacing the whole sprite with the split sprite at the respective position where the user would click on the whole sprite	Mahitha	Oct 12 – Oct 24
6	Movement of Half Sprites	This considers the movement of the split sprites outwards from the screen dimensions	Mahitha	Oct 25 – Oct 31
7	Sprite Rebounding	This implements the feature which ensures that the whole sprites move away with a slight increase in speed on colliding with each other.	Mahitha	Nov 1 – Nov 10
8	Timed Synchronization of Sprites	This ensures the synchronized appearance of Sprites on the screen on being cut	Mahitha	Nov 11 – Nov 21
9	Score Count	This feature is used to	Mahitha	Nov 13 –

		keep a track of the score		Nov 17
10	Time control of the game	This feature ensures the game display puts forward a timer which increments till 30 seconds and then the game exits	Mahitha	Nov 17 – Nov 21
11	Game Exit control	This feature implements the message that will be displayed once the game exits	Mahitha	Nov 22 – Nov 27
12	Game Integration	This involved merging of the game features after completion of independent tasks by the developers. This was done throughout the project timeline as and when required.	Mahitha	Oct 14 – Nov 27
13	Unit Testing/Bug Fixes	This was done to check the game worked as planned and there are no loop holes in the code	Mahitha	Oct 28 – Dec 1
14	Documentation	This task involves preparing the Project Completion Report and the 'Read Me' file.	Mahitha	Dec 2 – Dec 4

References

Web Links:

1) http://msdn.microsoft.com/en-us/library/bb200104.aspx
2) http://social.msdn.microsoft.com/Forums/en-US/xnagamestudioexpress/threads

Books:

1) Microsoft XNA Game Studio Creator's guide – An Introduction to XNA Game

Programming:
   a. Edition: 2007
   b. Authors: Stephen Cawood and Pat McGee
2) Learning XNA 4.0
   a. Edition: 2011
   b. Author: Aaron Reed

User Manual

1) Run the game
2) Gamer/User will view different objects/sprites moving across the screen and disappearing below the lower level X axis.
3) Gamer needs to use a mouse/touch screen to slice the objects by just clicking on the object. This increases the score by a value of 1.
4) If the user clicks a bomb, game ends displaying the score to the user
5) The game runs for a maximum of 30 seconds.
6) If the user does not click any bomb within the 30 seconds, the game will exit as said above displaying the score to the user.