

EDUCATIONAL GAME FOR MOBILE PLATFORMS

(Supervisor: Mr. Lo Hai Hiung)

by

Choong Meng Wei

(Student ID: 10637)

Final Report

December 2011

Electrical & Electronic Engineering Department

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

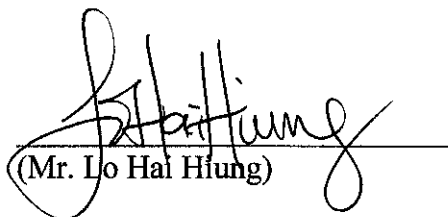
Educational Game For Mobile Platforms

by

Choong Meng Wei

A project dissertation submitted to the
Electrical and Electronic Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONIC ENGINEERING)

Approved by,



(Mr. Lo Hai Hiung)

Project Supervisor

Lo Hai Hiung
Lecturer
Electrical & Electronic Engineering
Universiti Teknologi PETRONAS
Bandar Seri Iskandar, 31750 Tronoh
Perak Darul Ridzuan, Malaysia

Universiti Teknologi PETRONAS

Tronoh, Perak

Dec 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



CHOONG MENG WEI

ABSTRACT

Gaming can be a very effective tool for learning. Game-based-learning has been studied as an important alternative or supplement to traditional teaching. With people using their mobile devices for gaming so often and with such a favorable demographic split amongst sex and age, the time to introduce mobile gaming as a learning tool may already be here. Mobile game can teach education materials in a more effective and interactive way. Mobile platforms can differ in terms of size, function, brand and operating system, hence to create a universal educational game engine for multiple mobile platforms is not an easy task. This project is aiming to create a mobile educational game from scratch. From graphic design to programing, from non existent to an actual functioning game. It describes the game engine design that has the potential to be expanded in the future.

ACKNOWLEDGEMENT

First of all, I would like to thank god for giving me the strength and health to do this project work until it done. Next I would like to express my greatest gratitude to my supervisor, Mr. Lo Hai Hiung. Without his assistance and guidance throughout the project, it is not possible for me to complete the project. We do face lots of difficulties during these two semesters, however, he patiently discuss the arising problems with me and give a clear vision for my project.

Besides that, I would like to thank my family member for providing me the support especially the crucial time of the project. For example, I really appreciate the Ipad that they gave to me as a birthday gift. Coincidentally, the Ipad has became one of my most important tools in this project. Their support and encouragement inspired me to complete the project in a much better way.

Last but not least, not forgotten my friends who were helping me in this project. We brainstormed and designed the puzzle together. They were helpful that when we discussed together, the problems became easily solved. Thank you to all of you for helping me throughout the tough period of the project.

TABLE OF CONTENTS

CERTIFICATION	II
ABSTRACT	IV
ACKNOWLEDGEMENT	V
LIST OF FIGURES	VIII
LIST OF TABLES	VIII
CHAPTER 1:	INTRODUCTION	1
	1.1 Overview	1
	1.2 Problem Statements	2
	1.3 Objectives	3
	1.4 Scope of Study	4
CHAPTER 2:	LITERATURE REVIEW	5
	2.1 Characteristics of Mobile Game	5
	2.2 Mobile Game Performance Optimization	6
	2.3 Multiplatform Support	7
	2.4 Educational Gameplay Design	8
	2.5 Others	9
CHAPTER 3:	METHODOLOGY	10
	3.1 Stages of The Project	10
	3.2 Marmalade SDK	13
	3.3 Software and Tools	15
	3.4 Gantt Chart	16

CHAPTER 4:	RESULTS AND DISCUSSION	17
4.1	Game Engine	17
4.2	Hamster Circuit	19
4.3	Gameplay	20
4.4	Deployment	22
4.5	Game Review	23
4.6	Potential Future Improvements.	23
CHAPTER 5:	CONCLUSION	24
CHAPTER 6:	REFERENCE	25
CHAPTER 7:	APPENDIX	26

LIST OF FIGURES

Figure 1: Marmalade SDK Simulator.....	10
Figure 2: Designing 2D Sprite.....	11
Figure 3: Project Activities.....	12
Figure 4: Marmalade SDK Application Development Workflow.....	14
Figure 5: Gantt Chart.....	16
Figure 6: Game level design and game engine.....	17
Figure 7: Level example for basic circuit theory.....	19
Figure 8: Level example for ladder logic.....	20
Figure 9: Start screen.....	20
Figure 10: Level-choosing screen.....	20
Figure 11: Example of tip sequence.....	21
Figure 12: Gameplay in Iphone.....	22
Figure 13: Gameplay in Ipad.....	22

LIST OF TABLES

Table 1: Game Engine Capabilities.....	18
Table 2: Future Potential Game Engine Capabilities.....	24

1.0 Introduction

1.1 Overview

Educational games are games that have been specifically designed to teach people about a certain subject, expand concepts, reinforce development, understand an historical event or culture, or assist them in learning a skill as they play. An educational mobile game can be defined as an electronic medium with all the characteristics of a gaming environment that have intended educational outcomes targeted at specific groups of learners.

This project is to design an educational mobile game using cross platform development kit named Marmalade SDK. With Marmalade SDK's deployment support for multiple mobile platforms, this game is aiming to reach out for massive audience. This project focuses on developing a game engine that can power different educational games with minimal interface design alterations. The main code of the game engine does not change with whatever game that is powered by it. The game engine is going to support most of the gameplay interactions such as multi-touch recognition, animation rendering, memory management etc. To demonstrate the capabilities of the game engine, a game named "Hamster Circuit" is designed for this project. This game is aiming to help gamers to learn the basic circuit theory while playing through the levels.

There are three phase of the project, at the first phase, the main task is to experiment the APIs in Marmalade SDK and build the game engine. Second phase's task is to design graphical and audio contents as well as the game levels for the game. The final phase is to test the game where it will be distributed to several game testers and refinements will be done according to the reviews received.

1.2 Problem Statements

The main problem that is faced by teachers nowadays is the ineffectiveness of teaching complicated subjects to student with the traditional teaching method. The exposure to the interactive media and entertainments has made the current generation students preferring interactive way of learning. Rapid development and introduction of advanced broadband access technologies, multimedia and mobile telecommunications services, are influencing the characteristics of contemporary students. Participation in virtual communities and the tendency to explore them seems quite natural to modern students, as they have grown up using Internet and mobile devices, through which they participate in standard communication channels such as SMS, e-mail, chat rooms, social communities, etc. They are more fond of multimedia and other Internet content than of traditional text books. Modern technology represents a way of life to them, rather than separate activities, which can be used for work and learning.

The described characteristics of the contemporary students require changes and adoption of innovative methods of teaching and learning. The traditional teaching approach which often involves only one way communication is no longer efficient to sustain students' interest especially when teaching the subjects that introduce a lot of theories and mind-drilling concepts.

Interactive teaching has been introduced since 1983 where the term "edutainment" was used to describe a package of software games for the Oric 1 and Spectrum Microcomputers in the UK. This approach has been proven to be more effective. However, it is not widely adopted by all the teachers. Sometimes, no matter how interesting or interactive the teaching style is, students might not be ready to absorb the knowledge. The readiness of students' minds is as important as the way of teaching.

Therefore, educational gaming is introduced to create a medium for spontaneous learning. Students can learn something new seamlessly through playing games. The concept of using games to change one's learning process from aggravation or even frustration to motivation is not a new one. It has been used with children to get them to learn a particularly difficult topic. With mobile game-based learning, students of all ages use the interactive gameplay system to creatively tackle complex problems. Competitive learning can be encouraged by enabling students to continue playing through the challenging yet rewarding mobile game levels.

Many people may have less time for those longer, more robust gaming experiences. Instead, they want the quick pick up and play nature games. Mobile games have made massive strides forward in terms of graphics and processing power. Additionally mobile games can be designed at a much lower cost than traditional PC or console games. Many gamers are now drawn to the mobile phone platform by the amazing technology and the relatively low prices in the application store.

1.3 Objective

The main objective of this project is to build a mobile game, where the game has to meet the following criteria:

1. Ability to provide interactive and educational gameplay experience
2. Ability to be deployed on several mobile platforms

Along with the main objectives, the game engine of the project should also be able to power other games in future with minimal gameplay design alterations.

1.4 Scope of Study

To specify the scope of the project, the mobile game will only be deployed onto IOS devices which include IPod, iPhone and IPad. With the free license that is provided by Marmalade SDK, one can only deploy and test their applications on IOS devices. To deploy onto other devices, premium license fee has to be paid. However, the game is capable of running on other devices that is powered by popular operating systems such as Android and Symbian. This option may be considered in future depends on how successful the gameplay design is.

The educational contents of the game will be related to Electrical and Electronics engineering field. The selected game title, “Hamster Circuit” will teach the gamers basic circuit theories. However, the game is targeting all audience disregarding their disciplines, ages and level of understanding of the subject.

2.0 Literature Review

2.1 Characteristics of Mobile Game

Mobile games are designed to utilize less graphical and computing power. Therefore, mobile game normally has simpler gameplay design compared to console game. The mobile games market has often been thought of as a refuge for independent game developers lacking the massive funds that have become necessary to produce console games [1]. Due to the limitation of scarce resources, such as memory, CPU, input and output, mobile game development is more difficult than desktop application development, with performance as one of the top critical requirements [2].

The game on mobile devices tends to be very different from PC game. One has to consider the shortcoming of mobile phone when design mobile game to suit for its characteristics. Anyway, the development of mobile games still meets the problems of compatibility and game development efficiency.

At present, most mobile phone companies commonly design several different mobile phones and each of them is pre-installed with several games, but games on different kinds of mobile phone are not compatible with each other. This problem makes programmers spend too much time in doing repeat work. Most mobile phone companies commonly develop their games on a given general SDK directly, which result in the slow development speed and low efficiency.

In order to develop mobile games in higher efficiency and reduce the burden of programmers, the popular method is to design game engine. By means of game engine, developers could design different kinds of game without considering the system details of the lower level. Therefore, the mobile phone companies are required to implement a common game development platform, which can improve the design

speed and make the game product run in any mobile phone that supports such platform [3]. A multiplatform mobile game can be sold in different application stores hence generates more profit compared to a game that is designed for only one specific platform.

2.2 Mobile Game Performance Optimization

There is a very critical requirement for good performance in spite of scarce resources on the mobile devices, such as low battery life, small memory and screen size, etc. Therefore, a good game may require special optimizations for certain hardware and software platforms. According to [4], the first step to improve the performance is considering methodology issues as they will have a huge influence on how to implement the system. More objects and classes will result in higher resource consumption. Traditional structural development is recommended [4]. Some optimizations resulting from this choice are:

1. Remove the constant interface.
2. Remove redundant inheritance relationships.
3. Remove unnecessary classes.
4. Remove obsolete class methods.
5. Find the performance bottleneck.

The second step of performance optimization is to minimize memory consumption and prevent memory leaks. Some of these techniques are classified as high level optimization and low level optimization. Code and Algorithm optimization may include: [4]

1. Do not initialize all objects in the constructor and initialize an object when first used

2. Change class members to local variables
3. Declare methods and variables final/static for faster access
4. Set an object to null if it is not used any more
5. Iterate loops down to zero
6. Avoid slow string comparisons.
7. Move the arithmetic operation in a loop outside the loop body
8. Use bitwise operators instead of multiplication and division

Datatype optimization is also an efficient way for improving performance as different data types use different resources [4].

1. Replace resizable Vectors with arrays, if possible.
2. Use appropriate data type and whenever possible use primitive data type instead of the object type.
3. Minimize the usage of string concatenation and comparison, use String Buffer instead.

2.3 Multiplatform Support

All mobile devices allow the development of native applications that can take advantage of specialized user interface elements and hardware components. While similar in capabilities, smartphones differ greatly in the way native applications have to be written for them [5].

The same differences exist in the APIs and programming models defined by Android, IOS, and Symbian. Although Android, the iPhone and the Symbian phones differ in their user interface style guidelines, but mobile games are an ideal candidate for cross-compilation. Games typically take over the complete screen and use few special purpose widgets. Since all platforms support almost identical animation and graphic capabilities, games can readily be cross-compiled [5].

Game developers prefer to use cross-platform tools to develop software in order to ensure their products are allowed being run on as many hardware platforms as possible [6]. One of the easiest methods of building a game that can compile and run on multiple platforms is using cross-platform game development tools like Marmalade SDK, Hereinto and Swerve Studio [6].

2.4 Educational Gameplay Design

Game-based learning may more adequately address the manner in which youngsters learn nowadays and engage them more successfully in meaningful learning than traditional learning methods [7]. Research has shown that educational games can provide sufficient stimulation to engage learners in knowledge discovery, while at the same time helping them to develop new skills [8].

The main characteristic that differentiates edutainment and video games is interactivity, because, the former being grounded on didactical and linear progressions, no place is left to wandering and alternatives [9]. A good educational mobile game should be deliberately designed to introduce non-simulated but fantasy elements to the gamers. While the game-play environment is conceived as a set of layers from game type, through narrative themes and interactions, to their tactical and strategic elements, hence it can inform the process of learning [10].

Research on mobile game-based learning tends to focus on the motivational effects of the methods. One of the main reasons for this is that when people play such games, they are generally found to be very engaged in the game: They are totally immersed in the game and can play for hours on end with little or no awareness of the more general world around them. Such a state of complete absorption in a task is referred to as 'flow' [11]. Flow theory is important for motivation in instruction. Having motivated learners is what many teachers desire. In fact, a motivated learner is easy to describe: enthusiastic, focused and engaged [11].

A motivated learner shows a clear interest in what he or she is doing and enjoys what he or she is doing, tries hard and persists over time. Many studies of games and motivation are based upon the motivation work of Malone and Lepper (1987) who proposed a link between motivation and intrinsic learning. More specifically, seven factors which include both individual and interpersonal factors have been postulated to promote intrinsic motivation. The individual factors are challenge, curiosity, control, fantasy, competition, cooperation and recognition. According to many authors, many of these factors are triggered by games [11].

2.5 Others

Since Marmalade SDK uses C++ as main programming language, knowing the best practices while using C++ programming in game design is useful. There are several good coding practices such as keeping the code simple, validating the inputs and considering the hardware restrictions [12].

The game is related to circuit theory, therefore reviewing the course is a good way to start the game design. Before designing the game, several popular games are played to learn their gameplay design as well as the unique interface they use to interact with gamers.

3.0 Methodology

3.1 Stages of The Project

This project can be divided into five stages. In the first stage, the main task is to experiment the API that is provided by Marmalade SDK. The C-style API, known as S3E (Segundo Embedded Execution Environment) behaves identically across all target platforms, thus reducing device and OS fragmentation. It does this by using a combination of a single application binary (the same for all platforms) and a thin C API which abstracts all relevant core OS services.

Marmalade supports an x86 build and debug environment, which is intended to be the primary means of development. It allows applications to be built and debugged without ever being sent to a device. It provides a much faster debug turn-around cycle than continually deploying to and testing on devices. Marmalade SDK supports integration with other development IDE. Program is debugged with the simulator that is provided by Marmalade SDK.

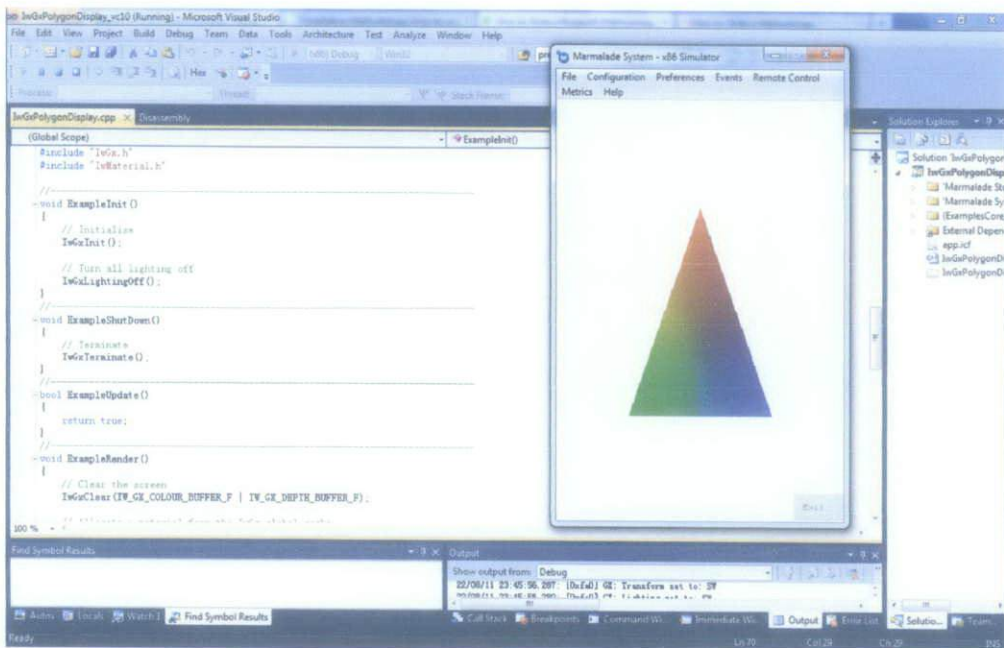


Figure 1: Marmalade SDK Simulator

After experimenting with all the useful API, next task is to build the game engine. This stage is the most important stage of the project because the main objective of this project is to build a game engine that can power several educational games. The capabilities of the game engine define the potentials of the game. The game engine of this project is designed to be able to adjust game specification according to devices, recognize touch interactions, manage memory and cpu resource, display animations and etc. These basic capabilities must be well established before moving forwards to the game designing phase because these features will affect the performance of the game greatly. Second stage's task is to design graphical and audio contents as well as the game levels for the game.

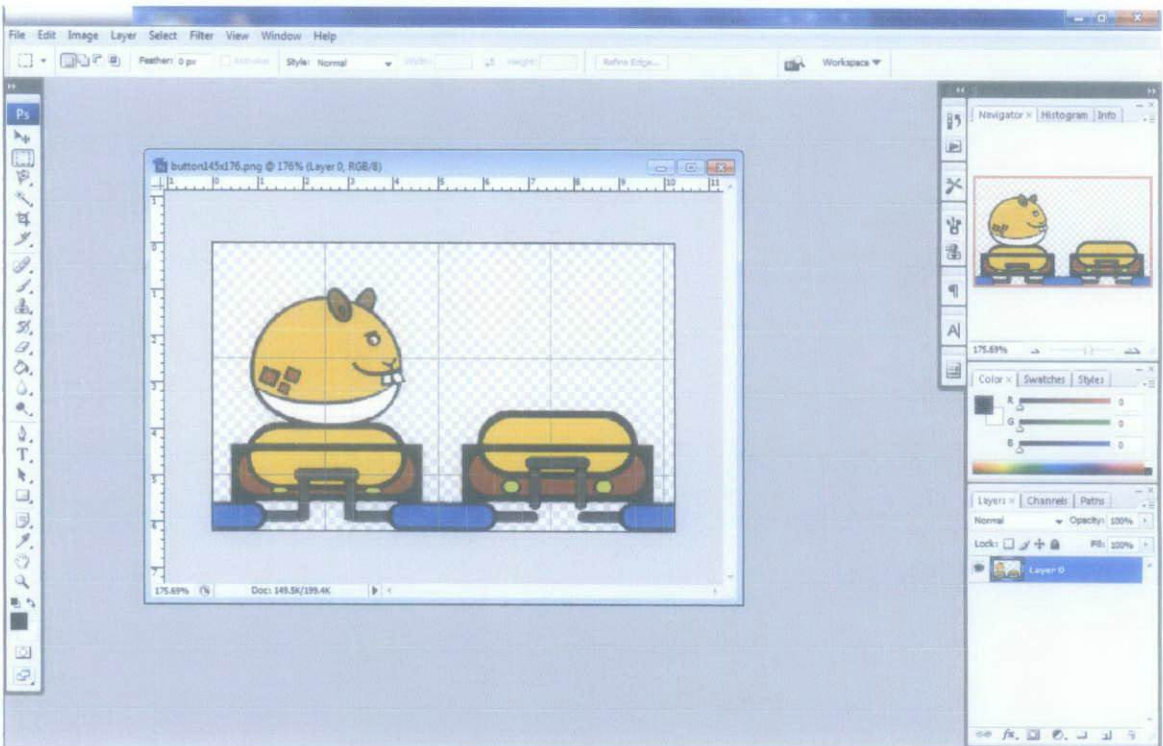


Figure 2: Designing 2D Sprite

With the game engine and graphical contents ready, game levels can be designed. The game is needed to test the game engine, therefore, the game is designed to be simple because the more advanced and complicated the game is, the more time is needed to build and debug it, which in this project, the objective is not to build a complicated game.

The final stage is the testing phase of the project, which the game will be distributed to several game testers and refinements will be done according to the reviews received. Game testers will test the performance of the game, help finding bugs and review the gameplay experience.

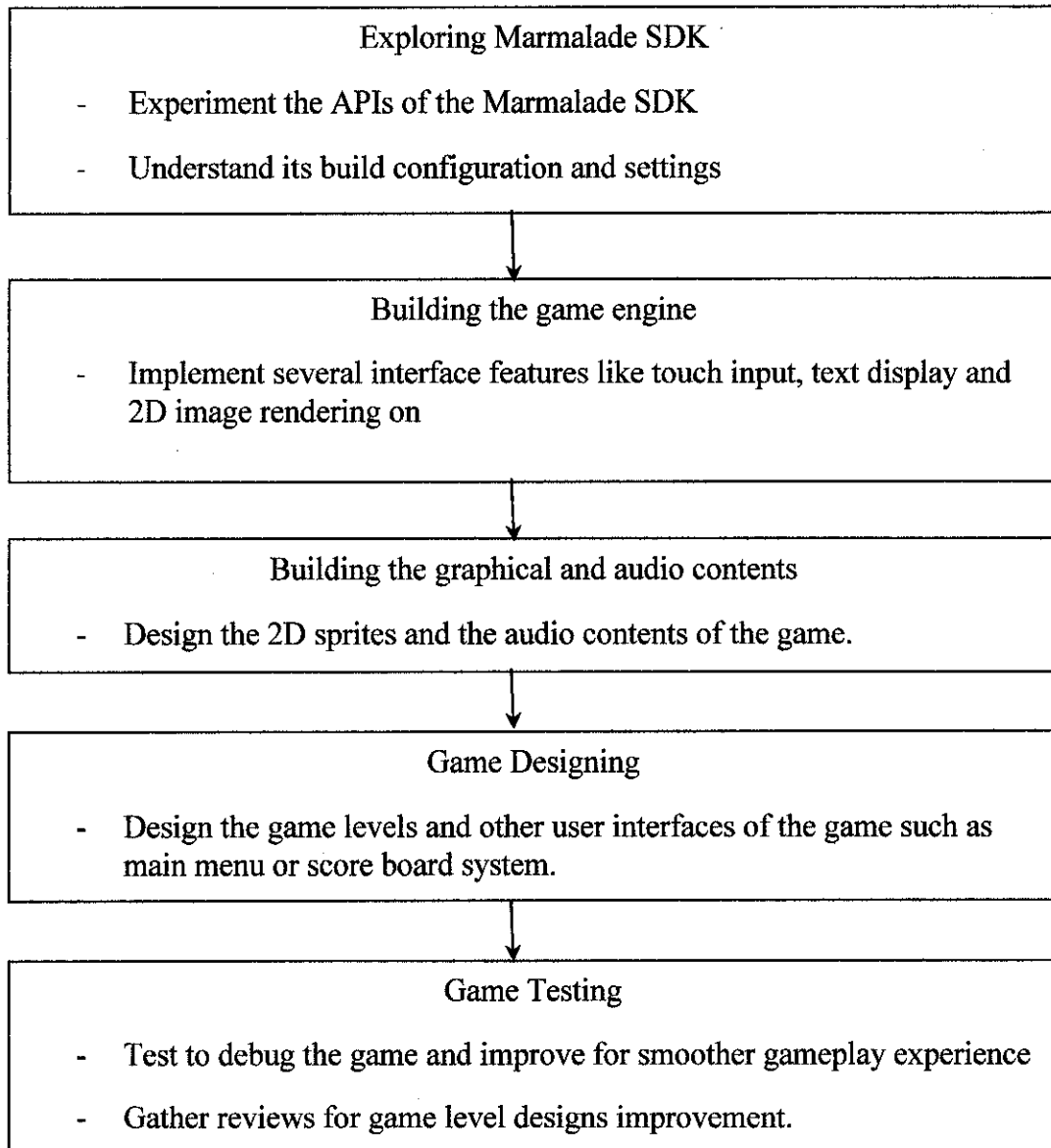


Figure 3: Project Activities

3.2 Marmalade SDK

Marmalade SDK is used in this project because it offers a complete solution for the creation of high-performance, multi-platform, native (C/C++) applications for mobile, embedded and desktop platforms [13]. All information about Marmalade SDK is cited from its documentation which is available in their official website.

Marmalade SDK is made up of a set of C-based APIs, a selection of freestanding tools, and plugins for various applications. These are linked together with a collection of specialised file types which allow data to be used, stored, and passed between the different processes. These elements, combined with a tailored workflow, allow applications to be planned, created, built and deployed with all the advantages that Marmalade brings [13].

Marmalade SDK consists of two major components: Marmalade System and Marmalade Studio. Marmalade System is an operating system abstraction API, together with the associated runtime libraries and application build system. It provides a series of tools (integrated with standard programming IDEs) and a C API to fully abstract the developer from the underlying target device OS [13].

Marmalade System can be thought of as a "virtual platform" that enables the programmer to write platform independent applications which run on all Marmalade-supported operating systems [13].

Marmalade Studio is a suite of tools and runtime components, focused on but not restricted to high-performance 2D/3D graphics and animation. Modules are provided to facilitate fixed-point and floating-point geometry, 2D and 3D drawing, model rendering, skinned animation, bitmap fonts, and more. Marmalade System includes support for fast software rendering, OpenGL ES 1.x and OpenGL ES 2.0. Marmalade

System functionality is provided as C++ APIs. Marmalade System "sits on top of" the Marmalade System OS-abstraction APIs, and thus all Marmalade System functionality is completely OS-independent [13].

The Application Development Workflow is the process used to create an application, from the creation of artwork, through coding, testing and debugging, to the deployment of the application onto a device. Each part of this process is crucial to creating an application and may often be carried out by many different people.

The image below shows a simplified version of the process for creating an application. The art assets are converted using the Marmalade Studio tools and added into the application built using the Marmalade SDK. This creates a single binary that can be used by the deployment tool to create an application to run on any of the different platforms [13].

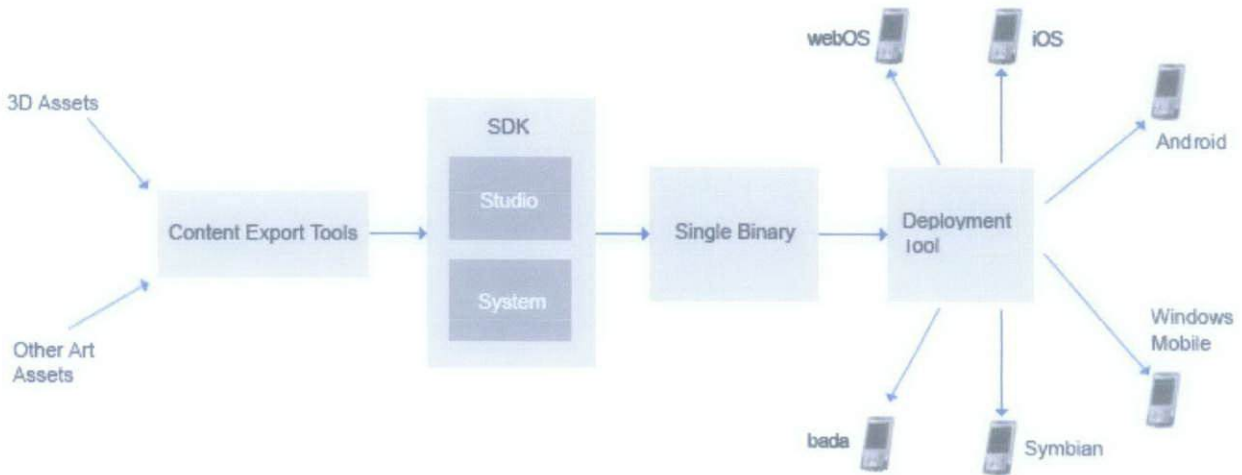


Figure 4: Marmalade SDK Application Development Workflow

3.2 Software And Tools

Marmalade SDK is free for evaluation purposes. In this project, the game is built for evaluation without the need of publishing, hence it is legal to use the SDK with the free licence. To publish the game and deploy it to other platform other than IOS devices, the premium licence fee has to be paid. All other software and hardware such as IOS devices are personal purchases and they are not added to the project expenses.

3.3 Gantt Chart

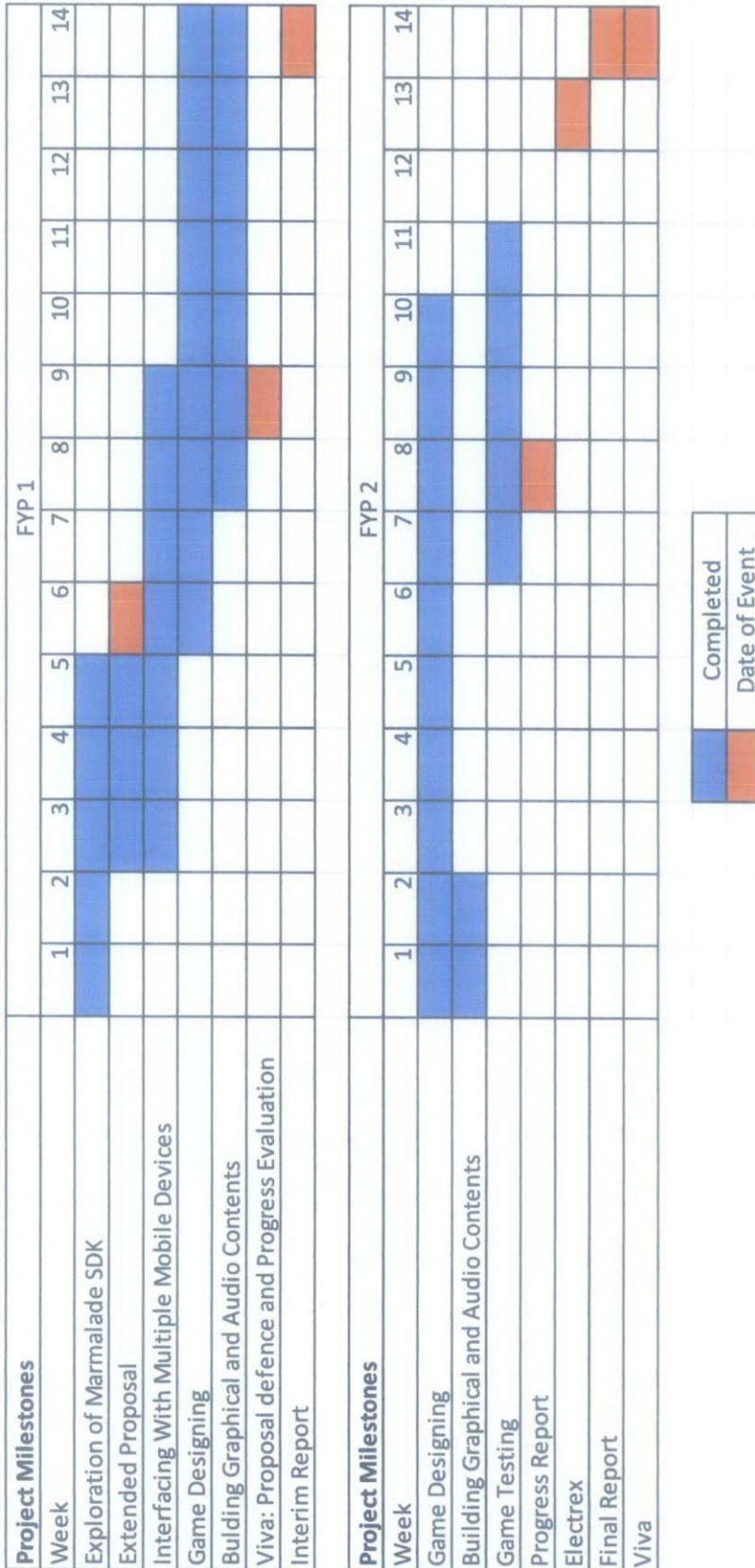


Figure 5: Gantt Chart

4.0 Result And Discussion

4.1 Game Engine

In order to make the game engine expandable and able to be easily adapted by other game in future, the code of the game is separated into two part:

- 1) Game Level Design
- 2) Game Engine

Game level design focuses on the level designs, the interactions in the game and everything else on how the game will be shown to gamers. On the other hand, game engine focuses on functions that is used to build the stage such as how the image is drawn, how the touch is registered, how the animation is generated. Game engine can be reused in future when developing other games. The code can be found on the appendix.

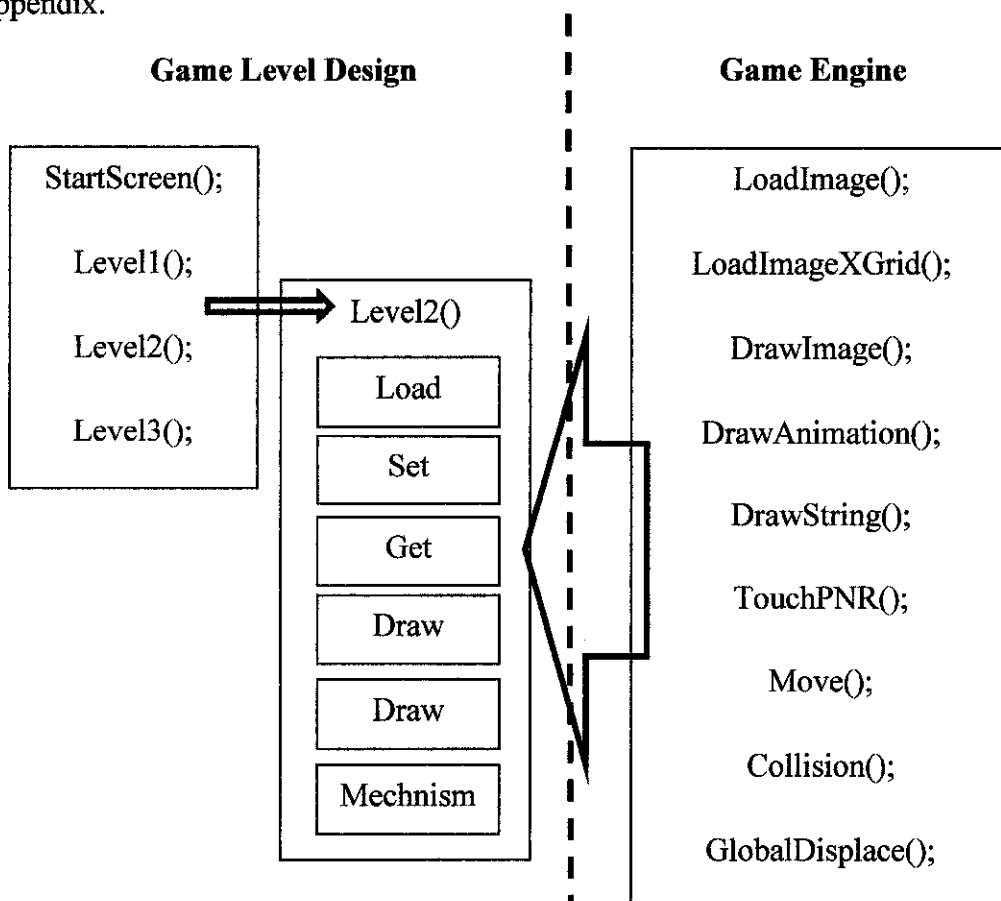


Figure 6: Game level design and game engine

Due to the time limit, there are about 13 levels designed in this project. It is sufficient to show the game mechanism and test the game engine. The table shows the capabilities of the game engine.

Capability	Description
Multitouch Recognition	<ul style="list-style-type: none"> - Recognise multiple touches on the screen and tag each touch with a unique ID - Recognise the gesture of each touch, whether the touches are sliding, holding, or released.
Image Display	<ul style="list-style-type: none"> - Display images on the screen with various features such as the capabilities of resizing, transforming and changing the depth.
Animation Rendering	<ul style="list-style-type: none"> - Display a stack of images according to the frame rate.
String Display	<ul style="list-style-type: none"> - Display text content on the screen with configurable fonts and alignments.
Sound	<ul style="list-style-type: none"> - Play sound from raw file and also other media formats
Collision Detection	<ul style="list-style-type: none"> - Detect collision between objects on the screen.
Scrolling	<ul style="list-style-type: none"> - Display image or text in scrolling animation and response to touch to control the scrolling speed.

Table 1: Game Engine Capabilities

4.2 Hamster Circuit

To test the features of the game engine, a game named Hamster Circuit is designed. This game is chosen from several titles that are suggested. Given the time limit of this project, the game should not be too complicated in term of the gameplay and interface. Hamster Circuit is a simple game that allows gamers to learn the basic circuit theory and ladder logic. In the beginning stages of the game, gamers are required to arrange the placement of each component including the hamster in order to make the circuit works. There are two theories that will be introduced in this game: basic circuit theory and ladder logic. In the stage of basic circuit theory, gamers can learn how circuit works. Some basic circuit concepts will be introduced such as considerations of the positive and negative poles of the components as well as the series and parallel connections.

Ladder logic is a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware. In this part of the game, gamers are required to arrange the buttons in the circuit in order to satisfy the equation that is given at the beginning of each level.

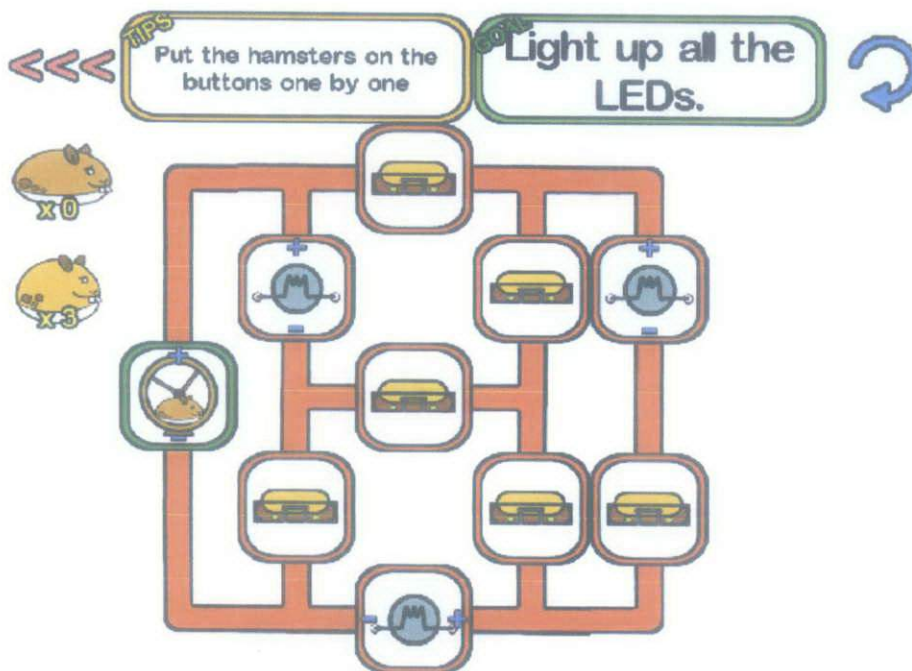


Figure 7: Level example for basic circuit theory

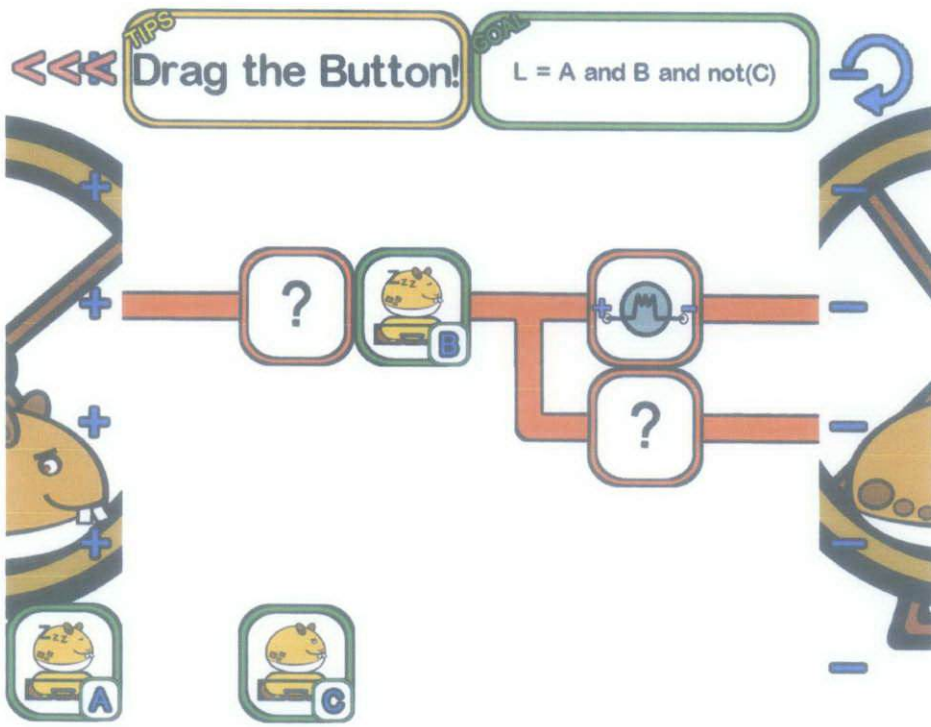


Figure 8: Level example for ladder logic

4.3 Gameplay

Hamster circuit is a puzzle themed game that teaches gamers about the circuit theory. Each theory will be introduced in a series of levels. The gameplay for each theory will ease gamers in the beginning from describing the most simple concept to encourage gamers in making complicated assumptions when the game progresses.

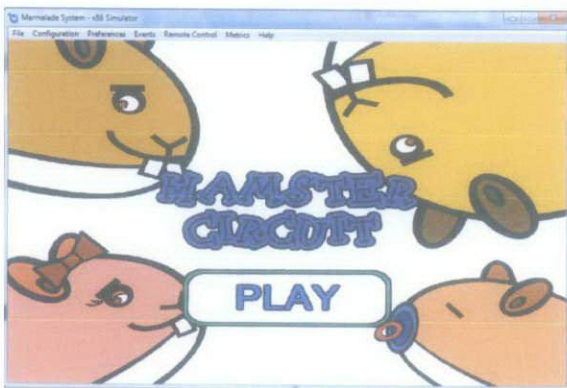


Figure 9: Start screen



Figure 10: Level-choosing screen

In each level, gamers are basically required to drag hamsters to the components in the circuit to achieve certain goals in order to advance to the next level. Provided in the left hand corner of the puzzle is an exclamation mark symbol which will prompt users to refer for tips. The tips will guide player through each level and will only appear depending on the status of the level. There are specific goals in each level. Gamers have to achieve each stated goal in order to proceed to the next level. Figure 10 shows how the tips are scrolled in a simple level.

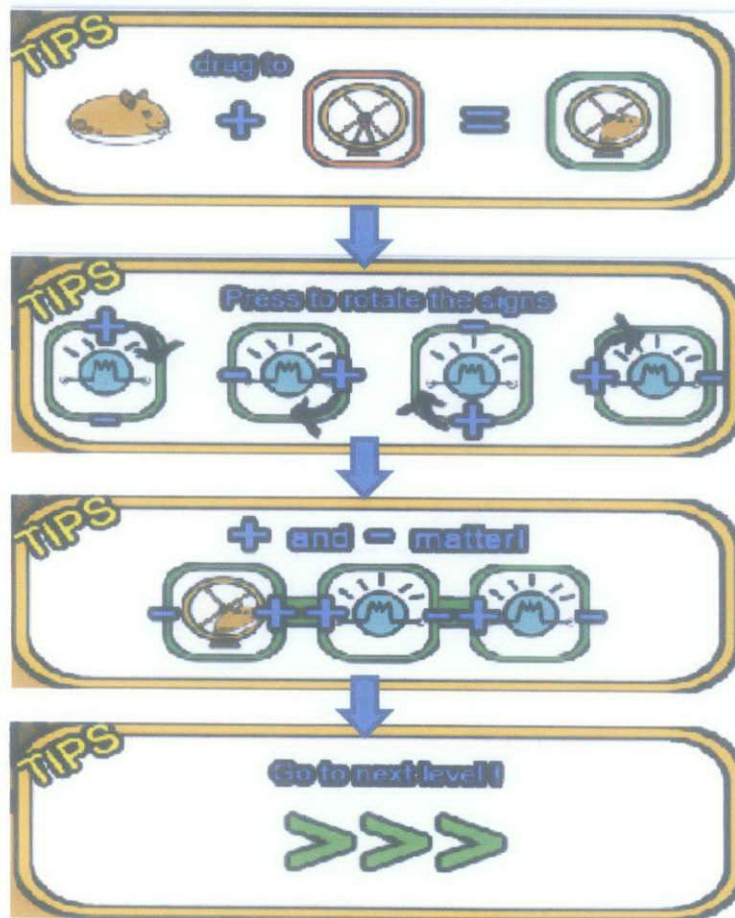


Figure 11: Example of tip sequence

4.4 Deployment

The game design in Ipad is slightly different from the game design in Iphone. Due to the smaller screen size in iphone, some informations have to be hidden in order to have more space to show the puzzle. The puzzles in both devices are the same.

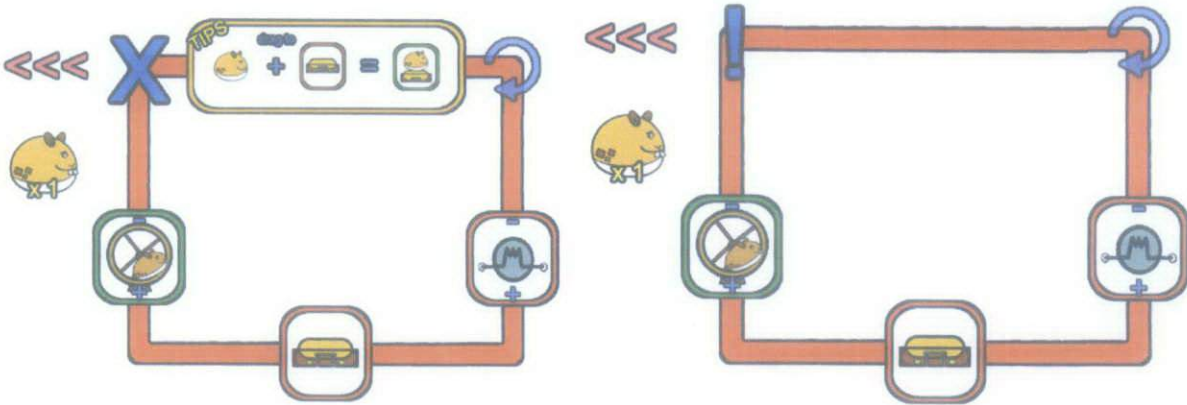


Figure 12: Gameplay in Iphone

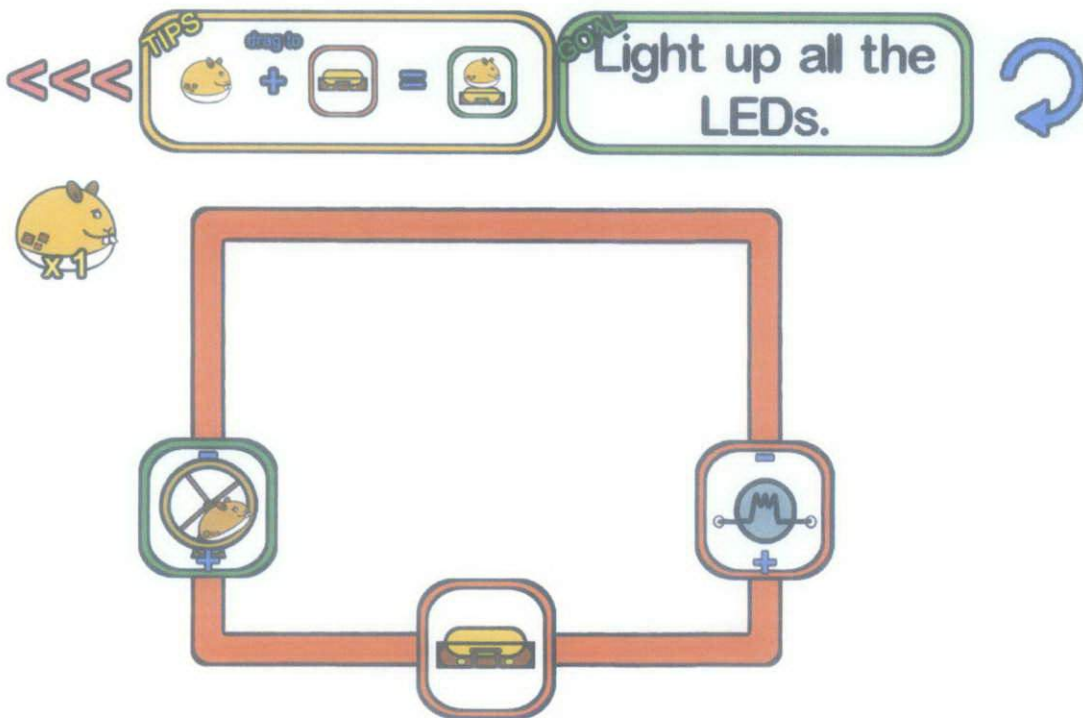


Figure 1: Gameplay in Ipad

Since image size of the game in Ipad is bigger, it requires more memory to render the gameplay. In order to avoid laggy gameplay experience, a fixed amount of memory is allocated before the game starts. For example, 25mb of memory is allocated for the iphone and 100mb is allocated for the Ipad.

4.5 Game Reviews

The game was tested by several students and some reviews are noted for future improvement. The overall review on the game performance is good. The interaction and the animation of the game are smooth and simple. However, different reviews are received on the game level designs. For the student who are studying electrical and electronics engineering, they commented that the puzzles are easy. For students from other courses, they said they need more description on the theories.

Some commented that the limited number of levels makes the game seem incomplete, hence it is incapable of explaining the theories in detail. Generally they get the direction that the game is trying to go, but to make the game more complete, more levels should be designed.

4.6 Potential Future Improvements

There are several features that were designed in this project but they are not well refined to be published due to the time limit. These features can be improved in future so that it can add more capabilities to the game engine. The future potential capabilities of the game engine are listed in the table below.

Capability	Description
Adjustable Depth	<ul style="list-style-type: none"> - Adjust the depth of the displayed images so that it creates 3D effect - Resize the image according to the depth
Collision Detection for Irregular Shaped Object	<ul style="list-style-type: none"> - Detect the edges of the object for collision detection for irregular shaped object.

Table 2: Future Potential Game Engine Capabilities

5.0 Conclusion

The objective to build a game engine that can power different educational games is met. The developed game engine supports most of the gameplay interactions such as multi-touch recognition, animation rendering, memory management and etc. “Hamster Circuit” is designed for this project to demonstrate the capabilities of the game engine. The game was tested by several students and some reviews are noted for future improvement. The overall review on the game performance is good. The interaction and the animation of the game are smooth and simple. Some commented that the limited number of levels makes the game seem incomplete, hence it is incapable of explaining the theories in detail. For future expansion and continuation, more levels should be designed and more features can be added to the game engine to improve its capabilities.

6.0 References

- [1] Ola Davidsson, J. P. (2006). Game Design Pattern For Mobile Games. *Game Design Pattern For Mobile Games*, 3.
- [2] Weishan Zhang, T. K. (2007). Mobile Game Development: Object-Orientation or Not. *Mobile Game Development: Object-Orientation or Not*, 1.
- [3] BIBLIOGRAPHY Bailin Yang, Z. Z. (2010). Design and Implementation of High Performance Mobile Game On Embedded Devices. *Design and Implementation of High Performance Mobile Game On Embedded Devices*, 2.
- [4] Weishan Zhang, T. K. (2007). Object-Orientation in Mobile Game. *Mobile Game Development: Object-Orientation or Not*, 4.
- [5] Yoon, A. P. (2010). Smartphone Cross-Compilation Framework for Multiplayer Online Games. *Smartphone Cross-Compilation Framework for Multiplayer Online Games*, 2.
- [6] Xin, C. (2009). Cross-Platform Mobile Phone Game Development Environment. *Cross-Platform Mobile Phone Game Development Environment*, 2.
- [7] J. Huizenga, * . A. (2009). Mobile game-based learning in secondary education: engagement, motivation and learning in a mobile city game. *Mobile game-based learning in secondary education: engagement, motivation and learning in a mobile city game*, 4.
- [8] Deepank Gupta, M. T. (2009). FoodForce2 - An Interactive and Collaborative Educational Learning Platform. *FoodForce2 - An Interactive and Collaborative Educational Learning Platform*, 4.
- [9] Dondlinger, M. J. (2007). Educational Video Game Design: A Review of the Literature. *Educational Video Game Design: A Review of the Literature*, 2.
- [10] O'Dea, M. (2008). Conceptual Development through Gameplay. *Conceptual Development through Gameplay*, 2.
- [11] Goth, G.S.(14 March 2005). Mobile learning with a mobile game:Design and Motivational Effects. Mobile learning with a mobile game:Design and Motivational Effects,4.
- [12] Dawson, M. (2004). *Beginning C++ Game Programming (Premier Press Game Development)* . Course Technology PTR.
- [13] Marmalade SDK Documentation,
<http://www.madewithmarmalade.com/devnet/docs>

7.0 Appendix

7.1 Headers

7.1.1 gamemain.h

```
#ifndef GEMEMAIN_H
#define GEMEMAIN_H

#include "s3e.h"
#include "Iw2D.h"
#include "IwGx.h"

#include "Touch.h"
#include "Object.h"
#include "Draw.h"

#include "gameglobal.h"
#include "stage1.h"

#endif //GEMEMAIN_H
```

7.1.2 Object.h

```
#ifndef OBJECT_H
#define OBJECT_H

//includes
#include "s3e.h"
#include "Iw2D.h"

#include "Touch.h"
#include "gameglobal.h"

//classes
class Object
{
private:

    //Touch interactions
    int32 objectID;
    bool collide;
    CIw2DImageTransform TRANS;
    bool PRESSED;
    bool PRESSALWAYSINREGION;
    CIwSVec2 SaveTouch;

    //Frames
    int32 FrameS;
    int64 t;
    CIw2DImage *sprite;
    bool IMAGE;

public:
```

```
//Dimension
CIwSVec2 Topleft;
CIwSVec2 CPos;
CIwSVec2 OCPos;
CIwSVec2 OriPos;
CIwSVec2 Size;
CIwSVec2 OSize;
int32 column;

//Data
bool KeyEvent[5];
int32 Counter[5];
int64 TimeOut[5];
ETOUCH TouchMech;

Object();
~Object();

void Update();
void LoadImageXGrid(const char* filename);
void LoadImageXGrid(const char* filename, CIwSVec2 Size);
void LoadImage(const char* filename);
void LoadImage(const char* filename, CIwSVec2 FSize);

void DrawImage(int32 Frame);
void DrawImage(int32 Frame, CIwSVec2 Pos);
void DrawImage(int32 Frame, CIwSVec2 Pos, POSITION POS);
void DrawImage(int32 Frame, CIwSVec2 Pos, POSITION POS, CIw2DImageTransform
trans);

void DrawScrollingImage(int32 Frame, CIwSVec2 Pos, CIwSVec2 TopLeftOffSet,
CIwSVec2 TopRightOffSet);
void DrawTopLeftImage(int32 Frame, CIwSVec2 Pos, int16 Ratio);

void DrawNumber(int32 Value, CIwSVec2 CenterPoint, CIw2DFont *FONT, int32 Colour);

void DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS,
CIw2DFont *FONT, int32 Colour);
void DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS,
CIw2DFont *FONT);
void DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS);
void DrawString(const char* String, CIwSVec2 CenterPoint);

void DrawStringCentered(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE);
void DrawStringCenteredBottom(const char* String, CIwSVec2 TopLEFT, CIw2DFont
*FONT, int32 Colour, CIwSVec2 TEXTREGIONSIZE);
void DrawStringTopLeft(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE);
void DrawStringCenteredTop(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE);

ETOUCH PNR();
ETOUCH PNR(CIwSVec2 ToPLeFT, CIwSVec2 SiZe);

void Position(CIwSVec2 Coor, POSITION POS);
```

```
void Displace(bool DisplaceX, bool DisplaceY);
void GlobalDisplacement();
void Move(DIRECTION Dir);
void DrawAnimation(char const* FrameSequence, int32 Ratems);
void DrawAnimation(char const* FrameSequence, int32 Ratems, CIwSVec2 TOpLeFt);

private:
    CDEBUG DEBUG;

};

#endif //OBJECT_H
```

7.1.3 Touch.h

```
#ifndef TOUCH_H
#define TOUCH_H

#include "s3e.h"
#include "Object.h"
#include "Iw2D.h"

#include "gameglobal.h"

//Struct
struct ATouch
{
    int32 x; // position x
    int32 y; // position y
    bool active; // is touch active (currently touching?)
    int32 id; // touch's unique identifier
    uint8 objectID; //the touching objectID
};

ATouch* ReturnSingleTouch();
ATouch* GetTouch(int32 id);
void SingleTouchMotionCB(s3ePointerMotionEvent* event);
void MultiTouchButtonCB(s3ePointerTouchEvent* event);
void MultiTouchMotionCB(s3ePointerTouchMotionEvent* event);
void SingleTouchButtonCB(s3ePointerEvent* event);
void TouchInit();
void TouchTerm();
void TouchUpdate();

#endif //TOUCH_H
```

7.1.4 Stage.h

```
#ifndef STAGE1_H
#define STAGE1_H

#include "s3e.h"
#include "Iw2D.h"
#include "IwGx.h"

#include "Touch.h"
#include "Object.h"
#include "Draw.h"

#include "gameglobal.h"

enum WIREMODE
{
    CTL,
    CTR,
    CBL,
    CBR,
    HOR,
    VER,
    TT,
    TL,
    TR,
    TB,
    X,
    VERM,
    HORM
};

class Cstage1
{
public:
    Cstage1 ();
    ~Cstage1 ();
    void Update();
    void Main();
    void StartScreen();
    void ClearData();
    void Refreshing(int16 ItemNumber);

    void ROLLInteraction(int16 i);
    void BUTTONInteraction(int16 i);
    void BUTTONDragging(int16 i, int16 QueI);
    void LEDInteraction(int16 i);
    void BUTTONCollision(int16 i);
    void ROLLCollision(int16 i);
    void QUEDrawing(int16 i);
    void LEDDrawing(int16 i, bool KeyEVENT);
    void HBDrawing();
    void HFDrawing();
    void ROLLDrawing(int16 i);
};
```

```
void BUTTONDrawing(int16 i);
void BUTTONDrawing(int16 i, const char* String);

void HBInteraction();
void HFInteraction();

void Level1();
void Level2();
void Level3();
void Level4();
void Level5();
void Level6();
void Level7();
void Level8();
void Level9();
void Level10();
void Level11();
void Level12();
void Level13();

void WIRING(int16 PosX, int16 PosY, WIREMODE SHAPE, int16 VOLTAGE);
bool Collision(CIwSVec2 OBJ1CPos, CIwSVec2 OBJ2CPos);
Object Drop(Object ObjDrag, Object ObjCol);
Object Drag(Object ObjDrag);
Object MOVE(Object OBJ);
void PlusMinus(int16 Counter);
void TipScrolling(char const* FrameSequence);
void Dialog(int16 TOPLX, int16 TOPLY, int16 WID, int16 HEI);

void SoundInit();

CIw2DFont *SmallFont, *MediumFont;
CDEBUG DEBUG;
int16 TIPS;
int16 LEVEL;
bool LEVELSTART;

private:
    Object O_HB, O_HF;
    Object O_ROLL[4];
    Object O_BUTTON[6];
    Object O_LED[6];
    Object O_QUE[6];
    Object O_RES[4];
    Object O_WIRE;

    Object O_BACK;
    Object O_NEXT;
    Object O_RESET;
    Object O_TIPS;
    Object O_ABC;
    Object O_GROUND;

    Object O_FRAME;
    Object O_CUR, O_INFO, O_PlusMinus;
    Object O_VOLT, O_HAM;
```

```
    Object O_CRITERIA2;

    Object O_END;
    Object O_TIPSFRAME;
    Object O_GOALFRAME;
    Object O_INFONUMBER;
    Object O_STAGE[10];
    Object O_PLAY, O_MENU;
    Object O_NUMBER;
    Object O_TITLE;
    Object O_FRAME2;
    Object O_DIALOG;
    Object O_STG1, O_STG2;
    Object O_POINTER;
    Object O_COMING;
    Object O_OK;
    Object O_WHAT;

    bool ITROLLRUN;
    bool ITBUTTONON;
    int16 infocounter;
    int64 timer, TipsTimer, ScrollTimer;
    int16 Scroll190;
    int16 TEMPFRAME;
};

#endif //STAGE1_H
```

7.2 Programs

7.2.1 HamsterCircuitIpad.cpp

```
#include "gamemain.h"
#include <malloc.h>

S3E_MAIN_DECL void IwMain()
{
    Iw2DInit();
    IwResManagerInit();
    Iw2DSetUseMipMapping(false);
    TouchInit();

    IwGetResManager()->LoadGroup("Data.group");

    while(1)
    {
        Iw2DSurfaceShow();
        IwGxClear(IW_GX_DEPTH_BUFFER_F);
        Iw2DSurfaceClear(0xFFFFFFFF);

        Cstapel* Gstapel = new Cstapel;
        Gstapel->Main();
        delete Gstapel;

        s3eDeviceYield(0);
        s3eKeyboardUpdate();
        s3ePointerUpdate();
        TouchUpdate();
        if(QUIT)break;
    }

    TouchTerm();
    IwResManagerTerminate();
    Iw2DTerminate();
}
```

7.2.2 Object.cpp

```
#include "Object.h"

int32 OBJECTID = 2;
CIwSVec2 OldPos = CIwSVec2(0,0);
CIwSVec2 NewPos = CIwSVec2(0,0);
CIwSVec2 Displacement;
bool SETPOS = false;

Object::Object()
{
    CPos = CIwSVec2(100,100);
}
```



```
    OCPos = CPos;
    OSize = CIwSVec2(100, 100);
    Size = OSize;
    Topleft = CIwSVec2(CPos.x-Size.x/2, CPos.y-Size.y/2);

    Counter[0] = 0;
    KeyEvent[0] = false;
    KeyEvent[1] = false;
    KeyEvent[2] = false;
    KeyEvent[3] = false;
    KeyEvent[4] = false;

    FrameS=0;
    PRESSED=false;
    PRESSALWAYSINREGION = true;
    SaveTouch = CIwSVec2(5555, 5555);
    IMAGE = false;
    TRANS = IW_2D_IMAGE_TRANSFORM_NONE;
    objectID = OBJECTID++;
}

Object::~~Object()
{
    if(IMAGE)
        delete sprite;
}

//Individual Object
void Object::LoadImageXGrid(const char* filename)
{
    sprite = Iw2DCreateImageResource(filename);
    IMAGE = true;
    OSize.x = sprite->GetWidth();
    OSize.y = sprite->GetHeight();
    Size = OSize;
    column = 1;
}

void Object::LoadImageXGrid(const char* filename, CIwSVec2 SiZe)
{
    sprite = Iw2DCreateImageResource(filename);
    IMAGE = true;
    OSize = SiZe;
    Size = OSize;
    column = sprite->GetWidth()/SiZe.x;
}

//Individual Object
void Object::LoadImage(const char* filename)
{
    sprite = Iw2DCreateImageResource(filename);
    IMAGE = true;
    OSize.x = SX;
    OSize.y = SY;
}
```

```
        Size = OSize;
        int32 TempWidth=sprite->GetWidth();
        column = TempWidth/OSize.x;
    }

void Object::LoadImage(const char* filename, CIwSVec2 FSize)
{
    sprite = Iw2DCreateImageResource(filename);
    int32 TempWidth=sprite->GetWidth();
    IMAGE = true;
    OSize = FSize;
    Size = OSize;
    column = TempWidth/OSize.x;
}

void Object::DrawImage(int32 Frame)
{
    Size = OSize;
    CIwSVec2 RegionOffset, RegionSize;

    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);
    RegionSize = OSize;
    Iw2DSetColour(WHITE);
    Iw2DDrawImageRegion(sprite, Topleft, Size, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}

void Object::DrawImage(int32 Frame, CIwSVec2 Pos)
{
    Size = OSize;
    Position(Pos, CENTER);

    CIwSVec2 RegionOffset, RegionSize;

    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);
    RegionSize = OSize;
    Iw2DSetColour(WHITE);
    Iw2DDrawImageRegion(sprite, Topleft, Size, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}

void Object::DrawImage(int32 Frame, CIwSVec2 Pos, POSITION POS)
{
    Size = OSize;
    Position(Pos, POS);

    CIwSVec2 RegionOffset, RegionSize;

    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);
    RegionSize = OSize;
    Iw2DSetColour(WHITE);
    Iw2DDrawImageRegion(sprite, Topleft, Size, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}
```

```
void Object::DrawImage(int32 Frame, CIwSVec2 Pos, POSITION POS, CIw2DImageTransform trans)
{
    Size = OSize;
    Position(Pos, POS);

    CIwSVec2 RegionOffset, RegionSize;

    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);
    RegionSize = OSize;
    Iw2DSetColour(WHITE);
    TRANS = trans;
    Iw2DSetImageTransform(TRANS);
    Iw2DDrawImageRegion(sprite, Topleft, Size, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}

void Object::DrawTopLeftImage(int32 Frame, CIwSVec2 Pos, int16 Ratio)
{
    CIwSVec2 Resize;
    Resize.x = Size.x*Ratio/100;
    Resize.y = Size.y*Ratio/100;
    CIwSVec2 RegionOffset, RegionSize;
    Position(Pos, TOPLEFT);
    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);
    RegionSize = OSize;
    Iw2DSetColour(WHITE);

    Iw2DDrawImageRegion(sprite, CIwSVec2(Pos.x + OSize.x/2 - Resize.x/2, Pos.y +
OSize.y/2 - Resize.y/2), Resize, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}

void Object::DrawScrollingImage(int32 Frame, CIwSVec2 Pos, CIwSVec2 TopLeftOffSet,
CIwSVec2 TopRightOffSet)
{
    CIwSVec2 RegionOffset, RegionSize;
    RegionOffset = CIwSVec2((Frame%(column))*OSize.x, (Frame/column)*OSize.y);

    RegionOffset.x = RegionOffset.x + TopLeftOffSet.x*OSize.x/100;
    RegionOffset.y = RegionOffset.y + TopLeftOffSet.y*OSize.y/100;

    RegionSize.x = OSize.x - TopRightOffSet.x*OSize.x/100-
TopLeftOffSet.x*OSize.x/100;
    RegionSize.y = OSize.y - TopRightOffSet.y*OSize.y/100-
TopLeftOffSet.y*OSize.y/100;

    Iw2DSetColour(WHITE);

    Iw2DDrawImageRegion(sprite, CIwSVec2(Pos.x + TopRightOffSet.x*OSize.x/100, Pos.y
+ TopRightOffSet.y*OSize.y/100), RegionSize, RegionOffset, RegionSize);
    Iw2DSetImageTransform(IW_2D_IMAGE_TRANSFORM_NONE);
}

void Object::DrawNumber(int32 Value, CIwSVec2 CenterPoint, CIw2DFont *FONT, int32 Colour)
{

```

```
    char DEC[3];
    sprintf(DEC, "%d", Value);
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);

    OSize.x = Iw2DGetStringWidth(DEC);
    OSize.y = OSize.x/3;
    Size = OSize;
    Iw2DDrawString(DEC, CIwSVec2(CenterPoint.x - Size.x/2, CenterPoint.y - Size.y/2),
    Size, IW_2D_FONT_ALIGN_CENTRE, IW_2D_FONT_ALIGN_CENTRE);
}
```

```
void Object::DrawStringCentered(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE)
{
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);
    Position(TopLEFT, TOPLEFT);
    Iw2DDrawString(String, TopLEFT, TEXTREGIONSIZE, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_CENTRE);
}
```

```
void Object::DrawStringCenteredBottom(const char* String, CIwSVec2 TopLEFT, CIw2DFont
*FONT, int32 Colour, CIwSVec2 TEXTREGIONSIZE)
{
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);
    Position(TopLEFT, TOPLEFT);
    Iw2DDrawString(String, TopLEFT, TEXTREGIONSIZE, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_BOTTOM);
}
```

```
void Object::DrawStringCenteredTop(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE)
{
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);
    Position(TopLEFT, TOPLEFT);
    Iw2DDrawString(String, TopLEFT, TEXTREGIONSIZE, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_TOP);
}
```

```
void Object::DrawStringTopLeft(const char* String, CIwSVec2 TopLEFT, CIw2DFont *FONT,
int32 Colour, CIwSVec2 TEXTREGIONSIZE)
{
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);
    Position(TopLEFT, TOPLEFT);
    Iw2DDrawString(String, TopLEFT, TEXTREGIONSIZE, IW_2D_FONT_ALIGN_LEFT,
IW_2D_FONT_ALIGN_TOP);
}
```

Copyright belongs to Choong Meng Wei

```
void Object::DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS, CIw2DFont
*FONT, int32 Colour)
{
    Iw2DSetFont(FONT);
    Iw2DSetColour(Colour);
    int32 CharCount=0;
    while(*String)
    {
        CharCount++;
        String++;
    }
    String = String - CharCount;

    OSize.x = Iw2DGetStringWidth(String);
    OSize.y = OSize.x/CharCount;
    Size = OSize;

    Position(CenterPoint, POS);
    Iw2DDrawString(String, Topleft, Size, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_CENTRE);
}
```

```
void Object::DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS, CIw2DFont
*FONT)
{
    Iw2DSetFont(FONT);
    int32 CharCount=0;
    while(*String)
    {
        CharCount++;
        String++;
    }
    String = String - CharCount;

    OSize.x = Iw2DGetStringWidth(String);
    OSize.y = OSize.x/CharCount;
    Size = OSize;

    Position(CenterPoint, POS);
    Iw2DDrawString(String, Topleft, Size, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_CENTRE);
}
```

```
void Object::DrawString(const char* String, CIwSVec2 CenterPoint, POSITION POS)
{
    int32 CharCount=0;
    while(*String)
    {
        CharCount++;
        String++;
    }
    String = String - CharCount;

    OSize.x = Iw2DGetStringWidth(String);
```

```

        OSize.y = OSize.x/CharCount;
        Size = OSize;

        Position(CenterPoint, POS);
        Iw2DDrawString(String, Topleft, Size, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_CENTRE);
    }

void Object::DrawString(const char* String, CIwSVec2 CenterPoint)
{
    int32 CharCount=0;
    while(*String)
    {
        CharCount++;
        String++;
    }
    String = String - CharCount;

    OSize.x = Iw2DGetStringWidth(String);
    OSize.y = OSize.x/CharCount;
    Size = OSize;

    Position(CenterPoint, CENTER);
    Topleft = CIwSVec2(CPos.x - Size.x/2, CPos.y - Size.y/2);
    Iw2DDrawString(String, Topleft, Size, IW_2D_FONT_ALIGN_CENTRE,
IW_2D_FONT_ALIGN_CENTRE);
}

ETOUCH Object::PNR()
{
    bool InRegion = ((Topleft.x <= ReturnSingleTouch()->x) && ((Topleft.x +
Size.x) >= ReturnSingleTouch()->x) && (Topleft.y <= ReturnSingleTouch()->y) &&
((Topleft.y +Size.y) >= ReturnSingleTouch()->y));
    if(ReturnSingleTouch()->active)
    {
        if(ReturnSingleTouch()->objectID == NULL)
        {
            if(InRegion)
            {
                PRESSED = true;
                ReturnSingleTouch()->objectID = objectID;
                SaveTouch = CIwSVec2(ReturnSingleTouch()-
>x, ReturnSingleTouch()->y);
                TouchMech = PRESSINREGION;
            }
            else
            {
                PRESSED = false;
                TouchMech = UNTOUCH;
            }
        }
        else if(ReturnSingleTouch()->objectID == objectID)
        {
            if(InRegion)
            {

```

```

        PRESSED = true;
        ReturnSingleTouch()->objectID = objectID;
        TouchMech = HOLDINREGION;
    }
    else
    {
        PRESSED = true;
        ReturnSingleTouch()->objectID = objectID;
        TouchMech = HOLDNOTINREGION;
    }
}
else //touching something else?
{
    PRESSED = false;
    TouchMech = UNTOUCH;
    return UNTOUCH;
}

}
else if(PRESSED) // releaseand previously press this object
{
    PRESSED = false;
    if(InRegion)
    {
        TouchMech = RELEASEINREGION;
    }
    else if(ReturnSingleTouch()->objectID == objectID)
    {
        TouchMech = RELEASENOTINREGION;
    }
    ReturnSingleTouch()->objectID = NULL;
}
else
{
    ReturnSingleTouch()->objectID = NULL;
    TouchMech = UNTOUCH;
}
return TouchMech;
}

ETOUCH Object::PNR(CIwSVec2 ToPLeFT, CIwSVec2 SiZe)
{
    bool InRegion = ((ToPLeFT.x <= ReturnSingleTouch()->x) && ((ToPLeFT.x +
SiZe.x) >= ReturnSingleTouch()->x) && (ToPLeFT.y <= ReturnSingleTouch()->y) &&
((ToPLeFT.y +SiZe.y) >= ReturnSingleTouch()->y));
    bool InRegionObject = ((Topleft.x <= ReturnSingleTouch()->x) && ((Topleft.x +
Size.x) >= ReturnSingleTouch()->x) && (Topleft.y <= ReturnSingleTouch()->y) &&
((Topleft.y +Size.y) >= ReturnSingleTouch()->y));
    if(ReturnSingleTouch()->active)
    {
        if(ReturnSingleTouch()->objectID == NULL)
        {
            if(InRegionObject)
            {
                PRESSED = true;
                ReturnSingleTouch()->objectID = objectID;
            }
        }
    }
}

```

```

        SaveTouch = CIwSVec2(ReturnSingleTouch()-
>x, ReturnSingleTouch()->y);
        TouchMech = PRESSINREGION;
    }
    else
    {
        PRESSED = false;
        TouchMech = UNTOUCH;
    }
}
else if(ReturnSingleTouch()->objectID == objectID)
{
    if(InRegion)
    {
        PRESSED = true;
        ReturnSingleTouch()->objectID = objectID;
        TouchMech = HOLDINREGION;
    }
    else
    {
        PRESSED = true;
        ReturnSingleTouch()->objectID = objectID;
        TouchMech = HOLDNOTINREGION;
    }
}
else //touching something else?
{
    PRESSED = false;
    TouchMech = UNTOUCH;
    return UNTOUCH;
}
}
else if(PRESSED) // releaseand previously press this object
{
    PRESSED = false;
    if(InRegion)
    {
        TouchMech = RELEASEINREGION;
    }
    else
    {
        TouchMech = RELEASENOTINREGION;
    }
    ReturnSingleTouch()->objectID = NULL;
}
else
{
    ReturnSingleTouch()->objectID = NULL;
    TouchMech = UNTOUCH;
}
return TouchMech;
}

```

```
void Object::DrawAnimation(char const* FrameSequence, int32 Ratems)
```



```
{
    DrawImage((int32) *(FrameSequence+FrameS)-48);
    if( (int32) (s3eTimerGetMs()-t)>Ratems)
    {
        if( *(FrameSequence + FrameS+1) !=NULL)
        {
            FrameS++;
        }
        else
        {
            FrameS = 0;
        }
        t = s3eTimerGetMs();
    }
}

void Object::DrawAnimation(char const* FrameSequence, int32 Ratems, CIwSVec2 TopLeFt)
{
    DrawImage((int32) *(FrameSequence+FrameS)-48, TopLeFt, TOPLEFT);
    if( (int32) (s3eTimerGetMs()-t)>Ratems)
    {
        if( *(FrameSequence + FrameS+1) !=NULL)
        {
            FrameS++;
        }
        else
        {
            FrameS = 0;
        }
        t = s3eTimerGetMs();
    }
}

void Object::Move(DIRECTION Dir)
{
    if(Dir == FORWARD)
    {
        CPos.x = OCPos.x + 10;
    }
    else if(Dir == BACKWARD)
    {
        CPos.x = OCPos.x - 10;
    }
}

//Global Object Manipulation Functions
void Object::GlobalDisplacement()
{
    if(PNR() == HOLDINREGION || PNR() == HOLDNOTINREGION)
    {
        SETPOS = false;
        if(NewPos == CIwSVec2(0, 0))
            OldPos = CIwSVec2(ReturnSingleTouch()->x, ReturnSingleTouch()->y);

        NewPos = CIwSVec2(ReturnSingleTouch()->x, ReturnSingleTouch()->y);
        Displacement = NewPos - OldPos;
    }
}
```

```
        OldPos = CIwSVec2(ReturnSingleTouch()->x, ReturnSingleTouch()->y);
    }
    else
    {
        //Initialise
        OldPos = CIwSVec2(0, 0);
        NewPos = CIwSVec2(0, 0);
        Displacement = CIwSVec2(0, 0);
    }

    DEBUG.StringWithVariable("The displacementX is %d", Displacement.x);
}
```

```
void Object::Position(CIwSVec2 Coor, POSITION POS)
```

```
{
    if(POS == TOPLEFT)
    {
        CPos.x = Coor.x + Size.x/2;
        CPos.y = Coor.y + Size.y/2;
    }
    else if(POS == CENTER)
    {
        CPos = Coor;
    }
    else if(POS == TOPRIGHT)
    {
        CPos.x = Coor.x - Size.x/2;
        CPos.y = Coor.y + Size.y/2;
    }
    else if(POS == BOTTOMRIGHT)
    {
        CPos.x = Coor.x - Size.x/2;
        CPos.y = Coor.y - Size.y/2;
    }
    else if(POS == BOTTOMLEFT)
    {
        CPos.x = Coor.x + Size.x/2;
        CPos.y = Coor.y - Size.y/2;
    }
    //Displace(1, 1);
    Topleft.x = CPos.x - Size.x/2;
    Topleft.y = CPos.y - Size.y/2;
}
```

```
void Object::Displace(bool DisplaceX, bool DisplaceY)
```

```
{

    if(Displacement==CIwSVec2(0, 0))
    {
        OCPos = CPos;
    }
    else
    {
        if(DisplaceX)
```

```
        CPos.x = OCPos.x + Displacement.x;
        if(DisplaceY)
            CPos.y = OCPos.y + Displacement.y;
    }
}

bool GCollide(CIwSVec2 ObjCPos, CIwSVec2 ObjOSize, CIwSVec2 Obj1CPos, CIwSVec2 Obj1OSize)
{
    int32 LbObj, RbObj, TbObj, BbObj, LbObj1, RbObj1, TbObj1, BbObj1;
    LbObj = ObjCPos.x - ObjOSize.x/2;
    RbObj = ObjCPos.x + ObjOSize.x/2;
    TbObj = ObjCPos.y - ObjOSize.y/2;
    BbObj = ObjCPos.y + ObjOSize.y/2;

    LbObj1 = Obj1CPos.x - Obj1OSize.x/2;
    RbObj1 = Obj1CPos.x + Obj1OSize.x/2;
    TbObj1 = Obj1CPos.y - Obj1OSize.y/2;
    BbObj1 = Obj1CPos.y + Obj1OSize.y/2;

    if( ( LbObj1 <= RbObj && LbObj1 >=LbObj ) && ( TbObj1 <= BbObj && TbObj1 >=
TbObj) ) // Top left corner overlapping?
        return true;
    if( ( RbObj1 <= RbObj && RbObj1 >=LbObj ) && ( TbObj1 <= BbObj && TbObj1 >=
TbObj) ) // Top right corner overlapping?
        return true;
    if( ( LbObj1 <= RbObj && LbObj1 >=LbObj ) && ( BbObj1 <= BbObj && BbObj1 >=
TbObj) ) // Bottom left corner overlapping?
        return true;
    if( ( RbObj1 <= RbObj && RbObj1 >=LbObj ) && ( BbObj1 <= BbObj && BbObj1 >=
TbObj) ) // Bottom right corner overlappin?
        return true;
    if( ( LbObj <= RbObj1 && LbObj >=LbObj1 ) && ( TbObj <= BbObj1 && TbObj >=
TbObj1) ) // Top left corner overlapping?
        return true;
    if( ( RbObj <= RbObj1 && RbObj >=LbObj1 ) && ( TbObj <= BbObj1 && TbObj >=
TbObj1) ) // Top right corner overlapping?
        return true;
    if( ( LbObj <= RbObj1 && LbObj >=LbObj1 ) && ( BbObj <= BbObj1 && BbObj >=
TbObj1) ) // Bottom left corner overlapping?
        return true;
    if( ( RbObj <= RbObj1 && RbObj >=LbObj1 ) && ( BbObj <= BbObj1 && BbObj >=
TbObj1) ) // Bottom right corner overlappin?
        return true;

    return false;
}
```

7.2.3 Touch.cpp

```
#include "Touch.h"

ATouch g_Touches[MAX_TOUCHES];

ATouch* ReturnSingleTouch()
{
    return &g_Touches[0];
}

//Find an active touch with the specified id, or allocate a free one from the list
ATouch* GetTouch(int32 id)
{
    ATouch* pInactive = NULL;

    for(uint32 i = 0; i < MAX_TOUCHES; i++)
    {
        if( id == g_Touches[i].id )
            return &g_Touches[i];
        if( !g_Touches[i].active )
            pInactive = &g_Touches[i];
    }

    //Return first inactive touch
    if( pInactive )
    {
        pInactive->id = id;
        return pInactive;
    }

    //No more touches, give up.
    return NULL;
}

void TouchInit()
{
    if (g_UseMultiTouch)
    {
        //Register for multi touch events on platforms where the functionality is
        //available.
        s3ePointerRegister(S3E_POINTER_TOUCH_EVENT, (s3eCallback)MultiTouchButtonCB,
        NULL);
        s3ePointerRegister(S3E_POINTER_TOUCH_MOTION_EVENT,
        (s3eCallback)MultiTouchMotionCB, NULL);
    }
    else
    {
        //Register for standard pointer events
        s3ePointerRegister(S3E_POINTER_BUTTON_EVENT, (s3eCallback)SingleTouchButtonCB,
        NULL);
        s3ePointerRegister(S3E_POINTER_MOTION_EVENT, (s3eCallback)SingleTouchMotionCB,
        NULL);
    }
}
```

```
void TouchTerm()
{
    //Clear up
    if (g_UseMultiTouch)
    {
        s3ePointerUnRegister(S3E_POINTER_TOUCH_EVENT, (s3eCallback)MultiTouchButtonCB);
        s3ePointerUnRegister(S3E_POINTER_TOUCH_MOTION_EVENT,
(s3eCallback)MultiTouchMotionCB);
    }
    else
    {
        s3ePointerUnRegister(S3E_POINTER_BUTTON_EVENT,
(s3eCallback)SingleTouchButtonCB);
        s3ePointerUnRegister(S3E_POINTER_MOTION_EVENT,
(s3eCallback)SingleTouchMotionCB);
    }
}

void TouchUpdate()
{
    if (g_UseMultiTouch)
    {
        s3ePointerUnRegister(S3E_POINTER_BUTTON_EVENT,
(s3eCallback)SingleTouchButtonCB);
        s3ePointerUnRegister(S3E_POINTER_MOTION_EVENT,
(s3eCallback)SingleTouchMotionCB);
        //Register for multi touch events.
        s3ePointerRegister(S3E_POINTER_TOUCH_EVENT, (s3eCallback)MultiTouchButtonCB,
NULL);
        s3ePointerRegister(S3E_POINTER_TOUCH_MOTION_EVENT,
(s3eCallback)MultiTouchMotionCB, NULL);
    }
    else
    {
        s3ePointerUnRegister(S3E_POINTER_TOUCH_EVENT, (s3eCallback)MultiTouchButtonCB);
        s3ePointerUnRegister(S3E_POINTER_TOUCH_MOTION_EVENT,
(s3eCallback)MultiTouchMotionCB);
        //Register for standard pointer events
        s3ePointerRegister(S3E_POINTER_BUTTON_EVENT, (s3eCallback)SingleTouchButtonCB,
NULL);
        s3ePointerRegister(S3E_POINTER_MOTION_EVENT, (s3eCallback)SingleTouchMotionCB,
NULL);
    }
}

void MultiTouchButtonCB(s3ePointerTouchEvent* event)
{
    ATouch* touch = GetTouch(event->m_TouchID);
    if (touch)
    {
        touch->active = event->m_Pressed != 0;
        touch->x = event->m_x;
        touch->y = event->m_y;
    }
}
```

```
void MultiTouchMotionCB(s3ePointerTouchMotionEvent* event)
{
    ATouch* touch = GetTouch(event->m_TouchID);
    if (touch)
    {
        touch->x = event->m_x;
        touch->y = event->m_y;
    }
}

void SingleTouchButtonCB(s3ePointerEvent* event)
{
    g_Touches[0].active = event->m_Pressed != 0;
    g_Touches[0].x = event->m_x;
    g_Touches[0].y = event->m_y;
}

void SingleTouchMotionCB(s3ePointerMotionEvent* event)
{
    g_Touches[0].x = event->m_x;
    g_Touches[0].y = event->m_y;
}
```

7.2.4 Stage1.cpp

```
#include "stage1.h"
Cstage1::Cstage1 ()
{
    IwGetResManager()->GetGroupNamed("Stage1");

    for(int i=0;i<10;i++)
    {
        O_STAGE[i].LoadImage("FRAME");
    }

    O_NUMBER.LoadImage("NUMBER");
    O_PLAY.LoadImageXGrid("PLAY");
    O_MENU.LoadImageXGrid("MENU");
    O_HB.LoadImage("HB");
    O_HF.LoadImage("HF");
    O_BUTTON[0].LoadImage("BUTTON");
    O_BUTTON[1].LoadImage("BUTTON");
    O_BUTTON[2].LoadImage("BUTTON");
    O_BUTTON[3].LoadImage("BUTTON");
    O_BUTTON[4].LoadImage("BUTTON");
    O_BUTTON[5].LoadImage("BUTTON");
    O_LED[0].LoadImage("LED");
    O_LED[1].LoadImage("LED");
    O_LED[2].LoadImage("LED");
    O_LED[3].LoadImage("LED");
    O_LED[4].LoadImage("LED");
    O_QUE[0].LoadImage("QUESTION");
    O_QUE[1].LoadImage("QUESTION");
    O_QUE[2].LoadImage("QUESTION");
}
```

```
O_QUE[3]. LoadImage("QUESTION");
O_QUE[4]. LoadImage("QUESTION");
O_ROLL[0]. LoadImage("ROLL");
O_WIRE. LoadImage("WIRE");
O_RES[0]. LoadImage("RESISTOR");
O_BACK. LoadImage("BACK");
O_PlusMinus. LoadImage("PLUSMINUS");
O_VOLT. LoadImage("VOLTMETER");
O_HAM. LoadImage("HAMMETER");
O_CRITERIA2. LoadImage("CRITERIA2");
O_TIPS. LoadImageXGrid("TIPS", CIwSVec2(3*SX, SY));
O_RESET. LoadImage("RESET");
O_NEXT. LoadImage("NEXT");
O_TIPSFRAME. LoadImageXGrid("TIPSFRAME");
O_GOALFRAME. LoadImageXGrid("GOALFRAME");
O_INFONUMBER. LoadImage("INFONUMBER");
O_TITLE. LoadImageXGrid("TITLE");
O_FRAME2. LoadImage("FRAME2");
//O_DIALOG. LoadImageXGrid("DIALOG");
O_STG1. LoadImageXGrid("STAGE1");
O_STG2. LoadImageXGrid("STAGE2");
O_COMING. LoadImageXGrid("COMING");
O_POINTER. LoadImage("POINTER");
O_OK. LoadImage("OK");
O_ABC. LoadImage("ABC");
O_GROUND. LoadImageXGrid("GROUND", CIwSVec2(SX, SY*6));

//Set font

IwGetResManager()->GetGroupNamed("Font");
MediumFont = Iw2DCreateFontResource("ArialR36");
SmallFont = Iw2DCreateFontResource("ArialR20");

LEVELSTART = true;
LEVEL = 0;
}

Cstage1::~Cstage1 ()
{
    delete SmallFont;
    delete MediumFont;
}

void Cstage1::Main()
{
    /*Play Background Music*/
    SoundInit();
    s3eAudioPlay("test.mp3", 0);

    while(1)
    {

        IwGxClear(IW_GX_DEPTH_BUFFER_F);
        Iw2DSurfaceClear(0xFFFFFFFF);
    }
}
```

```
switch (LEVEL)
{
case 0:
    {
        //Background Image
        StartScreen();
        break;
    }
case 1:
    {
        Level1();
        break;
    }
case 2:
    {
        Level2();
        break;
    }
case 3:
    {
        Level3();
        break;
    }
case 4:
    {
        Level4();
        break;
    }
case 5:
    {
        Level5();
        break;
    }
case 6:
    {
        Level6();
        break;
    }
case 7:
    {
        Level7();
        break;
    }
case 8:
    {
        Level8();
        break;
    }
case 9:
    {
        Level9();
        break;
    }
case 10:
    {
```



```
        Level10();
        break;
    }
    case 11:
    {
        Level11();
        break;
    }
    case 12:
    {
        Level12();
        break;
    }
    case 13:
    {
        Level13();
        break;
    }
    default:
    {
        StartScreen();
        break;
    }
}

Update();
if(QUIT)
{
    break;
}
}

void Cstage1::StartScreen()
{
    /*Start Screen*/
    if(O_BACK.Counter[0] == 0)
    {
        O_MENU.DrawImage(0, CIwSVec2(0, 0), TOPLEFT);
        O_PLAY.PNR();
        O_TITLE.DrawImage(0, CIwSVec2(SW/2, SH/2), CENTER);
        if(O_PLAY.TouchMech == RELEASEINREGION)
        {
            O_BACK.Counter[0] = 1;
        }
        O_PLAY.DrawImage(0, CIwSVec2(SX*2.5, SY*4), TOPLEFT);
    }
    /*level choosing*/
    else if(O_BACK.Counter[0] == 1)
    {
        O_STG1.KeyEvent[0]=false;
        O_STG2.KeyEvent[0]=false;
        O_COMING.KeyEvent[0]=false;
        O_BACK.PNR();

        if(O_BACK.TouchMech == RELEASEINREGION)
```

```

    {
        //back to start screen
        O_BACK.Counter[0]=0;
    }
    O_BACK.DrawImage(0, CIwSVec2(0, 0), TOPLEFT);

    O_STG1.DrawImage(0, CIwSVec2(SX, SY), TOPLEFT);
    O_MENU.DrawStringCenteredTop("CIRCUIT
THEORY", CIwSVec2(SX, SY*3), SmallFont, BLACK, CIwSVec2(SX*2, SY*2));
    if(O_STG1.PNR() == RELEASEINREGION)
    {
        O_STG1.KeyEvent[0]=true;
        O_BACK.Counter[0]=2;
    }
    O_STG2.DrawImage(0, CIwSVec2(SX*3.5, SY), TOPLEFT);
    O_MENU.DrawStringCenteredTop("LADDER
LOGIC", CIwSVec2(SX*3.5, SY*3), SmallFont, BLACK, CIwSVec2(SX*2, SY*2));
    if(O_STG2.PNR() == RELEASEINREGION)
    {
        O_STG2.KeyEvent[0]=true;
        O_BACK.Counter[0]=2;
    }
    O_COMING.DrawImage(0, CIwSVec2(SX*6, SY), TOPLEFT);
    O_MENU.DrawStringCenteredTop("Coming
Soon", CIwSVec2(SX*6, SY*3), SmallFont, BLACK, CIwSVec2(SX*2, SY*2));
    if(O_COMING.PNR() == RELEASEINREGION)
    {
        O_COMING.KeyEvent[0]=true;
        O_BACK.Counter[0]=2;
    }
}
else if(O_BACK.Counter[0] == 2)
{
    O_BACK.PNR();
    if(O_BACK.TouchMech == RELEASEINREGION)
    {
        O_BACK.Counter[0]=1;
    }
    O_BACK.DrawImage(0, CIwSVec2(0, 0), TOPLEFT);

    if(O_STG1.KeyEvent[0])
    {
        for(int16 i=0; i<7; i++)
        {
            O_STAGE[i].DrawImage(O_STAGE[i].KeyEvent[0], CIwSVec2(SX*(i%5+1), SY*(i/5+1)), TOP
LEFT);

            O_NUMBER.DrawTopLeftImage(i+1, O_STAGE[i].Topleft, 100);
            if(O_STAGE[i].PNR() == RELEASEINREGION)
            {
                LEVEL = i+1;
            }
        }
    }
}
else if(O_STG2.KeyEvent[0])

```

```

        {
            for(int16 i=0;i<6;i++)
            {
                O_STAGE[i].DrawImage(O_STAGE[i+7].KeyEvent[0],CIwSVec2(SX*(i%5+1),SY*(i/5+1)),T
OPLEFT);

                O_NUMBER.DrawTopLeftImage(i+1,O_STAGE[i].Topleft,100);
                if(O_STAGE[i].PNR() == RELEASEINREGION)
                {
                    LEVEL = i+8;
                }
            }
        }
        else if(O_COMING.KeyEvent[0])
        {
            O_MENU.DrawStringCenteredTop("THIS GAME IS IN TESTING
PHASE, MORE LEVELS WILL BE COMING SOON
. . .",CIwSVec2(SX*1,SY*2),MediumFont,BLACK,CIwSVec2(SX*6,SY*3));
        }
    }

}

void Cstage1::Level1()
{
    /*
        0   1   2   3   4   5   6   7
    0   BK   [   TIPS   ][   Goal   ] Rst
    1   HB   C           -           -           -           C
    2           |
    3           Br
    4           C           -           -           -           C
    5

    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_HB.Counter[0] = 1;
        O_ROLL[0].Counter[0] = 2;
        O_LED[0].Counter[0] = 2;
        /*Setting initial position*/
        O_HB.OriPos = CIwSVec2(0,SY);
        O_ROLL[0].OriPos = CIwSVec2(SX*1,SY*3);
        O_LED[0].OriPos = CIwSVec2(SX*5,SY*3);
        O_DIALOG.KeyEvent[0]=true;
    }

    //WIRING

```

```
WIRING(1, 1, CTL, O_WIRE. KeyEvent[0]);
WIRING(1, 2, VER, O_WIRE. KeyEvent[0]);
O_ROLL[0]. Position(CIwSVec2(SX*1, SY*3), TOPLEFT);
WIRING(1, 4, CBL, O_WIRE. KeyEvent[0]);

WIRING(2, 1, HOR, O_WIRE. KeyEvent[0]);
WIRING(2, 4, HOR, O_WIRE. KeyEvent[0]);
WIRING(3, 1, HOR, O_WIRE. KeyEvent[0]);
WIRING(3, 4, HOR, O_WIRE. KeyEvent[0]);
WIRING(4, 1, HOR, O_WIRE. KeyEvent[0]);
WIRING(4, 4, HOR, O_WIRE. KeyEvent[0]);

WIRING(5, 1, CTR, O_WIRE. KeyEvent[0]);
WIRING(5, 2, VER, O_WIRE. KeyEvent[0]);
O_LED[0]. Position(CIwSVec2(SX*5, SY*3), TOPLEFT);
WIRING(5, 4, CBR, O_WIRE. KeyEvent[0]);

/*Interaction when touch*/
HBInteraction();
ROLLCollision(0);

/*Puzzle Mechanism*/
if(O_ROLL[0]. KeyEvent[0])
{
    if((O_ROLL[0]. Counter[0] == 0 && O_LED[0]. Counter[0] == 0)
|| (O_ROLL[0]. Counter[0] == 2 && O_LED[0]. Counter[0] == 2))
    {
        O_WIRE. KeyEvent[0] = true;
    }
    else
    {
        O_WIRE. KeyEvent[0] = false;
    }
}
else
{
    O_WIRE. KeyEvent[0] = false;
}
O_STAGE[0]. KeyEvent[0] = O_WIRE. KeyEvent[0];

/*Draw components*/
LEDDrawing(0, O_WIRE. KeyEvent[0]);
ROLLDrawing(0);
HBDrawing();

//Check PNR
Refreshing(1);

if(!O_ROLL[0]. KeyEvent[0])
TipScrolling("0");
else if(!O_WIRE. KeyEvent[0])
TipScrolling("3");
else
TipScrolling("4");
```

```

O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

if(O_DIALOG.KeyEvent[0])
{
    Dialog(1,1,5,4);
    O_MENU.DrawStringCenteredTop("Level 1: Circuit
101",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
    O_ROLL[0].DrawTopLeftImage(2,CIwSVec2(SX*1.2,SY*2),100);
    O_MENU.DrawStringTopLeft("Roll generates
electricity.",CIwSVec2(SX*2.5,SY*2),MediumFont,BLACK,CIwSVec2(SX*4.3,SY*1));
    O_LED[0].DrawTopLeftImage(1,CIwSVec2(SX*1.2,SY*3),100);
    O_MENU.DrawStringTopLeft("LED lights up when there is
electricity.",CIwSVec2(SX*2.5,SY*3),MediumFont,BLACK,CIwSVec2(SX*4.3,SY*1));

    O_MENU.DrawStringCenteredTop("GOAL!",CIwSVec2(SX*1,SY*5),MediumFont,GREEN,CIwSV
ec2(SX*6,SY*1));
    O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

    O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));

    if(O_NEXT.PNR()==RELEASEINREGION)
    {
        O_DIALOG.KeyEvent[0] = false;
    }
}

void Cstage1::Level2()
{
    /*
        0  1  2  3  4  5  6  7
    0 BK
    1 HF    C    -    -    -    C
    2      |
    3      Br   L
    4      C    -    F    -    C
    5
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_HF.Counter[0] = 1;
        O_ROLL[0].KeyEvent[0] = true;
        O_ROLL[0].Counter[0] = 2;
        O_LED[0].Counter[0] = 2;
        /*Setting initial position*/
        O_HF.OriPos = CIwSVec2(0,SY);
    }
}

```

```
O_ROLL[0].OriPos = CIwSVec2(SX*1, SY*3);
O_LED[0].OriPos = CIwSVec2(SX*5, SY*3);
O_BUTTON[0].OriPos = CIwSVec2(SX*3, SY*4);
O_DIALOG.KeyEvent[0]=true;

}

//WIRING
WIRING(1, 1, CTL, O_WIRE.KeyEvent[0]);
WIRING(1, 2, VER, O_WIRE.KeyEvent[0]);
O_ROLL[0].Position(CIwSVec2(SX*1, SY*3), TOPLEFT);
WIRING(1, 4, CBL, O_WIRE.KeyEvent[0]);

WIRING(2, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(2, 4, HOR, O_WIRE.KeyEvent[0]);
WIRING(3, 1, HOR, O_WIRE.KeyEvent[0]);
O_BUTTON[0].Position(CIwSVec2(SX*3, SY*4), TOPLEFT);
WIRING(4, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(4, 4, HOR, O_WIRE.KeyEvent[0]);

WIRING(5, 1, CTR, O_WIRE.KeyEvent[0]);
WIRING(5, 2, VER, O_WIRE.KeyEvent[0]);
O_LED[0].Position(CIwSVec2(SX*5, SY*3), TOPLEFT);
WIRING(5, 4, CBR, O_WIRE.KeyEvent[0]);

/*Interaction when touch*/
HFInteraction();
BUTTONCollision(0);

/*Puzzle Mechanism*/
if(O_BUTTON[0].KeyEvent[0])
{
    if((O_ROLL[0].Counter[0] == 0 && O_LED[0].Counter[0] == 0)
|| (O_ROLL[0].Counter[0] == 2 && O_LED[0].Counter[0] == 2))
    {
        O_WIRE.KeyEvent[0] = true;
    }
    else
    {
        O_WIRE.KeyEvent[0] = false;
    }
}
else
{
    O_WIRE.KeyEvent[0] = false;
}
O_STAGE[1].KeyEvent[0] = O_WIRE.KeyEvent[0];

/*Draw components*/
BUTTONDrawing(0);
ROLLDrawing(0);
LEDDrawing(0, O_WIRE.KeyEvent[0]);
HFDrawing();

//Check PNR
```

```

Refreshing(1);

//TIPS Scrolling
if(!O_BUTTON[0].KeyEvent[0])
TipScrolling("6");
else if(!O_WIRE.KeyEvent[0])
TipScrolling("3");
else
TipScrolling("4");
O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

if(O_DIALOG.KeyEvent[0])
{
Dialog(1,1,5,4);
O_MENU.DrawStringCenteredTop("Level 2:
BUTTON?",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
O_BUTTON[0].DrawTopLeftImage(2,CIwSVec2(SX*1.2,SY*2),100);
O_MENU.DrawStringTopLeft("BUTTON connects the
circuit.",CIwSVec2(SX*2.5,SY*2),MediumFont,BLACK,CIwSVec2(SX*4.3,SY*1));

O_MENU.DrawStringCenteredTop("GOAL!",CIwSVec2(SX*1,SY*5),MediumFont,GREEN,CIwSV
ec2(SX*6,SY*1));
O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
if(O_NEXT.PNR()==RELEASEINREGION)
{
O_DIALOG.KeyEvent[0] = false;
}
}
}
void Cstage1::Level3()
{
/*
0 1 2 3 4 5 6 7
0 BK
1 HF C - TT - C
2 | B |
3 Br | L
4 C - TB - C
5
*/

if(LEVELSTART)
{
LEVELSTART = false;

ClearData();

/*Setting number of hamster, LED and ROLL initial plus minus
sign*/

```

```

    O_HF.Counter[0] = 0;
    O_ROLL[0].KeyEvent[0] = true;
    O_BUTTON[0].KeyEvent[0] = true;
    O_ROLL[0].Counter[0] = 2;
    O_LED[0].Counter[0] = 2;
    /*Setting initial position*/
    O_HF.OriPos = CIwSVec2(0, SY);
    O_ROLL[0].OriPos = CIwSVec2(SX*1, SY*3);
    O_LED[0].OriPos = CIwSVec2(SX*5, SY*3);
    O_BUTTON[0].OriPos = CIwSVec2(SX*3, SY*2);
    O_DIALOG.KeyEvent[0]=true;

}

//WIRING
WIRING(1, 1, CTL, O_WIRE.KeyEvent[0] || O_BUTTON[0].KeyEvent[0]);
WIRING(1, 2, VER, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);
O_ROLL[0].Position(CIwSVec2(SX*1, SY*3), TOPLEFT);
WIRING(1, 4, CBL, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);

WIRING(2, 1, HOR, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);
WIRING(2, 4, HOR, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);

WIRING(3, 1, TT, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);
WIRING(3, 3, VER, O_BUTTON[0].KeyEvent[0]);
O_BUTTON[0].Position(CIwSVec2(SX*3, SY*2), TOPLEFT);
WIRING(3, 4, TB, O_WIRE.KeyEvent[0] | O_BUTTON[0].KeyEvent[0]);

WIRING(4, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(4, 4, HOR, O_WIRE.KeyEvent[0]);

WIRING(5, 1, CTR, O_WIRE.KeyEvent[0]);
WIRING(5, 2, VER, O_WIRE.KeyEvent[0]);
O_LED[0].Position(CIwSVec2(SX*5, SY*3), TOPLEFT);
WIRING(5, 4, CBR, O_WIRE.KeyEvent[0]);

/*Interaction when touch*/
BUTTONInteraction(0);
HFInteraction();
BUTTONCollision(0);

/*Puzzle Mechanism*/
if(!O_BUTTON[0].KeyEvent[0])
{
    if((O_ROLL[0].Counter[0] == 0 && O_LED[0].Counter[0] == 0)
|| (O_ROLL[0].Counter[0] == 2 && O_LED[0].Counter[0] == 2))
    {
        O_WIRE.KeyEvent[0] = true;
    }
    else
    {
        O_WIRE.KeyEvent[0] = false;
    }
}
else
{

```



```

        O_WIRE.KeyEvent[0] = false;
    }
    O_STAGE[2].KeyEvent[0] = O_WIRE.KeyEvent[0];

    /*Draw components*/
    BUTTONDrawing(0);
    ROLLDrawing(0);
    LEDDrawing(0,O_WIRE.KeyEvent[0]);
    HFDrawing();

    //Check PNR
    Refreshing(1);

    //TIPS Scrolling
    if(O_BUTTON[0].KeyEvent[0])
        TipScrolling("7");
    else
        TipScrolling("4");
    O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

    if(O_DIALOG.KeyEvent[0])
    {
        Dialog(1,1,5,4);
        O_MENU.DrawStringCenteredTop("Level 3: Short
Circuit?",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
        O_BUTTON[0].DrawTopLeftImage(2,CIwSVec2(SX*1.2,SY*2),100);
        O_MENU.DrawStringTopLeft("BUTTON will also short the
circuit.",CIwSVec2(SX*2.5,SY*2),MediumFont,BLACK,CIwSVec2(SX*4.3,SY*1));

        O_MENU.DrawStringCenteredTop("GOAL!",CIwSVec2(SX*1,SY*5),MediumFont,GREEN,CIwSV
ec2(SX*6,SY*1));
        O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

        O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
        if(O_NEXT.PNR()==RELEASEINREGION)
        {
            O_DIALOG.KeyEvent[0] = false;
        }
    }
}
void Cstagal::Level4()
{
    /*
        0   1   2   3   4   5
    0 BK
    1 HB   C   -   -   -   C
    2     |
    3     Br
    4     C   -   -   -   C
    */

```

```
5
*/

if(LEVELSTART)
{
    LEVELSTART = false;

    ClearData();

    /*Setting number of hamster, LED and ROLL initial plus minus
sign*/
    O_ROLL[0].KeyEvent[0] = true;
    O_ROLL[0].Counter[0] = 2;
    O_LED[0].Counter[0] = 3;
    O_HB.Counter[0]=0;
    /*Setting initial position*/
    O_HB.OriPos = CIwSVec2(0,SY);
    O_ROLL[0].OriPos = CIwSVec2(SX*1,SY*3);
    O_LED[0].OriPos = CIwSVec2(SX*5,SY*3);
    O_DIALOG.KeyEvent[0]=true;
}

//WIRING
WIRING(1, 1, CTL, O_WIRE.KeyEvent[0]);
WIRING(1, 2, VER, O_WIRE.KeyEvent[0]);
O_ROLL[0].Position(CIwSVec2(SX*1,SY*3), TOPLEFT);
WIRING(1, 4, CBL, O_WIRE.KeyEvent[0]);

WIRING(2, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(2, 4, HOR, O_WIRE.KeyEvent[0]);
WIRING(3, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(3, 4, HOR, O_WIRE.KeyEvent[0]);
WIRING(4, 1, HOR, O_WIRE.KeyEvent[0]);
WIRING(4, 4, HOR, O_WIRE.KeyEvent[0]);

WIRING(5, 1, CTR, O_WIRE.KeyEvent[0]);
WIRING(5, 2, VER, O_WIRE.KeyEvent[0]);
O_LED[0].Position(CIwSVec2(SX*5,SY*3), TOPLEFT);
WIRING(5, 4, CBR, O_WIRE.KeyEvent[0]);

/*Interaction when touch*/
ROLLInteraction(0);
HBInteraction();
ROLLCollision(0);
LEDInteraction(0);

/*Puzzle Mechanism*/
if(O_ROLL[0].KeyEvent[0])
{
    if((O_ROLL[0].Counter[0] == 0 && O_LED[0].Counter[0] == 0)
|| (O_ROLL[0].Counter[0] == 2 && O_LED[0].Counter[0] == 2))
    {
        O_WIRE.KeyEvent[0] = true;
    }
}
```

```

        else
        {
            O_WIRE.KeyEvent[0] = false;
        }
    }
    else
    {
        O_WIRE.KeyEvent[0] = false;
    }
    O_STAGE[3].KeyEvent[0] = O_WIRE.KeyEvent[0];

    /*Draw components*/
    ROLLDrawing(0);
    HBDrawing();
    LEDDrawing(0, O_WIRE.KeyEvent[0]);

    //Check PNR
    Refreshing(1);

    //TIPS Scrolling
    if(O_ROLL[0].KeyEvent[0])
    {
        if(!O_WIRE.KeyEvent[0])
            TipScrolling("3");
        else
            TipScrolling("4");
    }
    else
        TipScrolling("0");
    O_MENU.DrawStringCentered("Light up all the
LEDs.", CIwSVec2(SX*4, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));

    if(O_DIALOG.KeyEvent[0])
    {
        Dialog(1, 1, 5, 4);
        O_MENU.DrawStringCenteredTop("Level 4: PLUS and
MINUS?", CIwSVec2(SX*1, SY*1.2), MediumFont, BLACK, CIwSVec2(SX*6, SY*0.8));
        O_TIPS.DrawTopLeftImage(1, CIwSVec2(SX*2.5, SY*2), 125);
        O_MENU.DrawStringCentered("Press and rotate those signs, \nRoll's
positive pole should be connected to LED's positive
pole.", CIwSVec2(SX*1, SY*3), SmallFont, BLACK, CIwSVec2(SX*6, SY*1));

        O_MENU.DrawStringCenteredTop("GOAL!", CIwSVec2(SX*1, SY*5), MediumFont, GREEN, CIwSV
ec2(SX*6, SY*1));
        O_MENU.DrawStringCentered("Light up all the
LEDs.", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

        O_NEXT.DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX
, SY));
        if(O_NEXT.PNR() == RELEASEINREGION)
        {
            O_DIALOG.KeyEvent[0] = false;
        }
    }

```

```

    }
}
void Cstage1::Level5()
{
    /*
        0   1   2   3   4   5   6   7
    0 BK
    1 HB   C   -   L   -   C
    2     |
    3     Br
    4     C   -   -   -   C
    5
    */

    if (LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_ROLL[0].KeyEvent[0] = true;
        O_ROLL[0].Counter[0] = 2;
        O_LED[0].Counter[0] = 3;
        O_HB.Counter[0]=0;
        /*Setting initial position*/
        O_HB.OriPos = CIwSVec2(0, SY);
        O_ROLL[0].OriPos = CIwSVec2(SX*1, SY*3);
        O_LED[0].OriPos = CIwSVec2(SX*3, SY*1);
        O_DIALOG.KeyEvent[0]=true;

    }

    //WIRING
    WIRING(1, 1, CTL, O_WIRE.KeyEvent[0]);
    WIRING(1, 2, VER, O_WIRE.KeyEvent[0]);
    O_ROLL[0].Position(CIwSVec2(SX*1, SY*3), TOPLEFT);
    WIRING(1, 4, CBL, O_WIRE.KeyEvent[0]);

    WIRING(2, 1, HOR, O_WIRE.KeyEvent[0]);
    WIRING(2, 4, HOR, O_WIRE.KeyEvent[0]);
    O_LED[0].Position(CIwSVec2(SX*3, SY*1), TOPLEFT);
    WIRING(3, 4, HOR, O_WIRE.KeyEvent[0]);
    WIRING(4, 1, HOR, O_WIRE.KeyEvent[0]);
    WIRING(4, 4, HOR, O_WIRE.KeyEvent[0]);

    WIRING(5, 1, CTR, O_WIRE.KeyEvent[0]);
    WIRING(5, 2, VER, O_WIRE.KeyEvent[0]);
    WIRING(5, 3, VER, O_WIRE.KeyEvent[0]);
    WIRING(5, 4, CBR, O_WIRE.KeyEvent[0]);

    /*Interaction when touch*/
    ROLLInteraction(0);
    HBInteraction();
}

```

```

ROLLCollision(0);
LEDInteraction(0);

/*Puzzle Mechanism*/
if(O_ROLL[0].KeyEvent[0])
{
    if((O_ROLL[0].Counter[0] == 0 && O_LED[0].Counter[0] == 3)
|| (O_ROLL[0].Counter[0] == 2 && O_LED[0].Counter[0] == 1))
    {
        O_WIRE.KeyEvent[0] = true;
    }
    else
    {
        O_WIRE.KeyEvent[0] = false;
    }
}
else
{
    O_WIRE.KeyEvent[0] = false;
}
O_STAGE[4].KeyEvent[0] = O_WIRE.KeyEvent[0];

/*Draw components*/
ROLLDrawing(0);
HBDrawing();
LEDDrawing(0,O_WIRE.KeyEvent[0]);

//Check PNR
Refreshing(1);

//TIPS Scrolling
if(O_ROLL[0].KeyEvent[0])
{
    if(!O_WIRE.KeyEvent[0])
        TipScrolling("3");
    else
        TipScrolling("4");
}
else
    TipScrolling("0");
O_MENU.DrawStringCentered("Light up all the
LEDs. ",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

if(O_DIALOG.KeyEvent[0])
{
    Dialog(1,1,5,4);
    O_MENU.DrawStringCenteredTop("Level 5: Practice Makes
Perfect!",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
    O_TIPS.DrawTopLeftImage(1,CIwSVec2(SX*2.5,SY*2),125);
    O_MENU.DrawStringCentered("Press and rotate those signs,\nRoll's
positive pole should be connected to LED's positive
pole. ",CIwSVec2(SX*1,SY*3),SmallFont,BLACK,CIwSVec2(SX*6,SY*1));

```

```

O_MENU.DrawStringCenteredTop("GOAL!", CIwSVec2(SX*1, SY*5), MediumFont, GREEN, CIwSVec2(SX*6, SY*1));
O_MENU.DrawStringCentered("Light up all the LEDs.", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

O_NEXT.DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX, SY));
    if(O_NEXT.PNR()==RELEASEINREGION)
    {
        O_DIALOG.KeyEvent[0] = false;
    }
}
void Cstage1::Level6()
{
    /*
        0   1   2   3   4   5   6   7
    0
    1   BK   C   TT   L1   TT   C
    2   HB   |   L0   |   L4
    3       Br   |   L3   |
    4       C   TB   L2   TB   C
    5
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus sign*/

        O_ROLL[0].KeyEvent[0] = true;
        O_ROLL[0].Counter[0] = 1;
        O_LED[0].Counter[0] = 3;
        O_LED[1].Counter[0] = 3;
        O_LED[2].Counter[0] = 3;
        O_LED[3].Counter[0] = 3;
        O_LED[4].Counter[0] = 3;
        O_HB.Counter[0]=0;
        /*Setting initial position*/
        O_HB.OriPos = CIwSVec2(0, SY);
        O_ROLL[0].OriPos = CIwSVec2(SX*1, SY*3);
        O_LED[0].OriPos = CIwSVec2(SX*2, SY*2);
        O_LED[1].OriPos = CIwSVec2(SX*3, SY*1);
        O_LED[2].OriPos = CIwSVec2(SX*3, SY*4);
        O_LED[3].OriPos = CIwSVec2(SX*4, SY*3);
        O_LED[4].OriPos = CIwSVec2(SX*5, SY*2);

        O_DIALOG.KeyEvent[0]=true;
    }
}

```

```

//WIRING

WIRING(1, 1, CTL, O_WIRE. KeyEvent[0] || O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);

WIRING(1, 2, VER, O_WIRE. KeyEvent[0] || O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);
    O_ROLL[0]. Position(CIwSVec2(SX*1, SY*3), TOPLEFT);

WIRING(1, 4, CBL, O_WIRE. KeyEvent[0] || O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);

WIRING(2, 1, TT, O_WIRE. KeyEvent[0] || O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);
    O_LED[0]. Position(CIwSVec2(SX*2, SY*2), TOPLEFT);
    WIRING(2, 3, VER, O_WIRE. KeyEvent[0]);

WIRING(2, 4, TB, O_WIRE. KeyEvent[0] || O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);

    O_LED[1]. Position(CIwSVec2(SX*3, SY*1), TOPLEFT);
    O_LED[2]. Position(CIwSVec2(SX*3, SY*4), TOPLEFT);
    WIRING(4, 1, TT, O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);
    WIRING(4, 2, VER, O_WIRE. KeyEvent[1]);
    O_LED[3]. Position(CIwSVec2(SX*4, SY*3), TOPLEFT);
    WIRING(4, 4, TB, O_WIRE. KeyEvent[1] || O_WIRE. KeyEvent[2]);

WIRING(5, 1, CTR, O_WIRE. KeyEvent[2]);
O_LED[4]. Position(CIwSVec2(SX*5, SY*2), TOPLEFT);
WIRING(5, 3, VER, O_WIRE. KeyEvent[2]);
WIRING(5, 4, CBR, O_WIRE. KeyEvent[2]);

/*Interaction when touch*/
ROLLInteraction(0);
HBInteraction();
ROLLCollision(0);
LEDInteraction(0);
LEDInteraction(1);
LEDInteraction(2);
LEDInteraction(3);
LEDInteraction(4);

/*Puzzle Mechanism*/
if(O_ROLL[0]. KeyEvent[0])
{
    if(O_ROLL[0]. Counter[0] == 0)
    {
        O_WIRE. KeyEvent[0] = (O_LED[0]. Counter[0] == 0)?
true:false;
        if(O_LED[1]. Counter[0] == 3 && O_LED[2]. Counter[0] == 1)
        {
            O_WIRE. KeyEvent[1] = (O_LED[3]. Counter[0] == 0)?
true:false;
            O_WIRE. KeyEvent[2] = (O_LED[4]. Counter[0] == 0)?
true:false;
        }
        else
        {
            O_WIRE. KeyEvent[1] = false;

```

```

        O_WIRE.KeyEvent[2] = false;
    }
}
else if(O_ROLL[0].Counter[0] == 2)
{
    O_WIRE.KeyEvent[0] = (O_LED[0].Counter[0] == 2)?
true:false;

    if(O_LED[1].Counter[0] == 1 && O_LED[2].Counter[0] == 3)
    {
        O_WIRE.KeyEvent[1] = (O_LED[3].Counter[0] == 2)?
true:false;

        O_WIRE.KeyEvent[2] = (O_LED[4].Counter[0] == 2)?
true:false;
    }
    else
    {
        O_WIRE.KeyEvent[1] = false;
        O_WIRE.KeyEvent[2] = false;
    }
}
else
{
    O_WIRE.KeyEvent[0] = false;
    O_WIRE.KeyEvent[1] = false;
    O_WIRE.KeyEvent[2] = false;
}
}
else
{
    O_WIRE.KeyEvent[0] = false;
    O_WIRE.KeyEvent[1] = false;
    O_WIRE.KeyEvent[2] = false;
}

O_STAGE[5].KeyEvent[0] =
O_WIRE.KeyEvent[0]&&O_WIRE.KeyEvent[1]&&O_WIRE.KeyEvent[2];

/*Draw components*/
ROLDDrawing(0);
HBDrawing();
LEDDrawing(0,O_WIRE.KeyEvent[0]);
LEDDrawing(1,O_WIRE.KeyEvent[1]||O_WIRE.KeyEvent[2]);
LEDDrawing(2,O_WIRE.KeyEvent[1]||O_WIRE.KeyEvent[2]);
LEDDrawing(3,O_WIRE.KeyEvent[1]);
LEDDrawing(4,O_WIRE.KeyEvent[2]);

//Check PNR
Refreshing(5);

//TIPS Scrolling
if(O_ROLL[0].KeyEvent[0])
{
    if(!O_WIRE.KeyEvent[0]||!O_WIRE.KeyEvent[1]||!O_WIRE.KeyEvent[2])
    TipScrolling("3");
}

```



```

        else
            TipScrolling("4");
    }
    else
        TipScrolling("0");
    O_MENU.DrawStringCentered("Light up all the
LEDs.", CIwSVec2(SX*4, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));

    if(O_DIALOG.KeyEvent[0])
    {
        Dialog(1, 1, 5, 4);
        O_MENU.DrawStringCenteredTop("Level 6: Complicated + and -
", CIwSVec2(SX*1, SY*1.2), MediumFont, BLACK, CIwSVec2(SX*6, SY*0.8));
        O_TIPS.DrawTopLeftImage(1, CIwSVec2(SX*2.5, SY*2), 125);
        O_MENU.DrawStringCentered("Press and rotate those signs, \nRoll's
positive pole should be connected to LED's positive
pole.", CIwSVec2(SX*1, SY*3), SmallFont, BLACK, CIwSVec2(SX*6, SY*1));

        O_MENU.DrawStringCenteredTop("GOAL!", CIwSVec2(SX*1, SY*5), MediumFont, GREEN, CIwSV
ec2(SX*6, SY*1));
        O_MENU.DrawStringCentered("Light up all the
LEDs.", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

        O_NEXT.DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX
, SY));
        if(O_NEXT.PNR()==RELEASEINREGION)
        {
            O_DIALOG.KeyEvent[0] = false;
        }
    }
}
void Cstage1::Level7()
{
    /*
        0   1   2   3   4   5   6   7
    0
    1   BK   C       TT   B0   TT   C
    2   HF   |       LO       B1   L2
    3   HB   Br   TL   B2   TR   B5
    4       |       B3       B4   |
    5       C   TB   L1   TB   C
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/
        O_ROLL[0].KeyEvent[0] = true;
        O_ROLL[0].Counter[0] = 1;
        O_LED[0].Counter[0] = 3;
    }
}

```

```

O_LED[1].Counter[0] = 3;
O_LED[2].Counter[0] = 3;
O_HF.Counter[0]=3;
O_HB.Counter[0]=0;
/*Setting initial position*/
O_HB.OriPos = CIwSVec2(0,SY);
O_HF.OriPos = CIwSVec2(0,SY*2);
O_ROLL[0].OriPos = CIwSVec2(SX*1,SY*3);
O_LED[0].OriPos = CIwSVec2(SX*2,SY*2);
O_LED[1].OriPos = CIwSVec2(SX*3,SY*5);
O_LED[2].OriPos = CIwSVec2(SX*5,SY*2);
O_BUTTON[0].OriPos = CIwSVec2(SX*3,SY*1);
O_BUTTON[1].OriPos = CIwSVec2(SX*4,SY*2);
O_BUTTON[2].OriPos = CIwSVec2(SX*3,SY*3);
O_BUTTON[3].OriPos = CIwSVec2(SX*2,SY*4);
O_BUTTON[4].OriPos = CIwSVec2(SX*4,SY*4);
O_BUTTON[5].OriPos = CIwSVec2(SX*5,SY*4);

O_DIALOG.KeyEvent[0]=true;

}

//WIRING
WIRING(1,1,CTL,O_WIRE.KeyEvent[0]);
WIRING(1,2,VER,O_WIRE.KeyEvent[0]);
O_ROLL[0].Position(CIwSVec2(SX*1,SY*3),TOPLEFT);
WIRING(1,4,VER,O_WIRE.KeyEvent[0]);
WIRING(1,5,CBL,O_WIRE.KeyEvent[0]);

WIRING(2,1,TT,O_WIRE.KeyEvent[0]);
O_LED[0].Position(CIwSVec2(SX*2,SY*2),TOPLEFT);
WIRING(2,3,TL,O_WIRE.KeyEvent[0]);
O_BUTTON[3].Position(CIwSVec2(SX*2,SY*4),TOPLEFT);
WIRING(2,5,TB,O_WIRE.KeyEvent[0]);

O_BUTTON[0].Position(CIwSVec2(SX*3,SY*1),TOPLEFT);
O_BUTTON[2].Position(CIwSVec2(SX*3,SY*3),TOPLEFT);
O_LED[1].Position(CIwSVec2(SX*3,SY*5),TOPLEFT);

WIRING(4,1,TT,O_WIRE.KeyEvent[0]);
O_BUTTON[1].Position(CIwSVec2(SX*4,SY*2),TOPLEFT);
O_BUTTON[4].Position(CIwSVec2(SX*4,SY*4),TOPLEFT);
WIRING(4,3,TR,false);
WIRING(4,5,TB,O_WIRE.KeyEvent[0]);

WIRING(5,1,CTR,O_WIRE.KeyEvent[0]);
O_BUTTON[5].Position(CIwSVec2(SX*5,SY*4),TOPLEFT);
O_LED[2].Position(CIwSVec2(SX*5,SY*2),TOPLEFT);
WIRING(5,3,VER,O_WIRE.KeyEvent[0]);
WIRING(5,5,CBR,O_WIRE.KeyEvent[0]);

/*Interaction when touch*/
ROLLInteraction(0);
HBInteraction();
HFInteraction();

```

```
ROLLCollision(0);
for(int i=0;i<6;i++)
{
    BUTTONInteraction(i);
    BUTTONCollision(i);
}
LEDInteraction(0);
LEDInteraction(1);
LEDInteraction(2);

/*Puzzle Mechanism*/
if(O_ROLL[0].KeyEvent[0])
{
    if(O_ROLL[0].Counter[0] == 0)
    {
        O_LED[0].KeyEvent[0] = (O_LED[0].Counter[0] == 0)?
true:false;
        O_LED[1].KeyEvent[0] = (O_LED[1].Counter[0] == 1)?
true:false;
        O_LED[2].KeyEvent[0] = (O_LED[2].Counter[0] == 0)?
true:false;
    }
    else if(O_ROLL[0].Counter[0] == 2)
    {
        O_LED[0].KeyEvent[0] = (O_LED[0].Counter[0] == 2)?
true:false;
        O_LED[1].KeyEvent[0] = (O_LED[1].Counter[0] == 3)?
true:false;
        O_LED[2].KeyEvent[0] = (O_LED[2].Counter[0] == 2)?
true:false;
    }
    else
    {
        O_LED[0].KeyEvent[0] = false;
        O_LED[1].KeyEvent[0] = false;
        O_LED[2].KeyEvent[0] = false;
    }

    if( O_BUTTON[3].KeyEvent[0]&&
        O_BUTTON[0].KeyEvent[0]&&
        O_BUTTON[5].KeyEvent[0]&&
        O_LED[0].KeyEvent[0]&&
        O_LED[1].KeyEvent[0]&&
        O_LED[2].KeyEvent[0])
    {
        O_WIRE.KeyEvent[0] = true;
    }
    else
    {
        O_WIRE.KeyEvent[0] = false;
    }
}
else
{
    O_WIRE.KeyEvent[0] = false;
}
```

```

    }

    O_STAGE[6].KeyEvent[0] = O_WIRE.KeyEvent[0];

    /*Draw components*/
    ROLLDrawing(0);
    for(int i=0;i<6;i++)
    BUTTONDrawing(i);
    LEDDrawing(0,O_WIRE.KeyEvent[0]);
    LEDDrawing(1,O_WIRE.KeyEvent[0]);
    LEDDrawing(2,O_WIRE.KeyEvent[0]);
    HBDrawing();
    HFDrawing();

    //Check PNR
    Refreshing(6);

    //TIPS Scrolling
    if(O_ROLL[0].KeyEvent[0])
    {
        if(    O_LED[0].KeyEvent[0]&&
            O_LED[1].KeyEvent[0]&&
            O_LED[2].KeyEvent[0])
        {
            O_MENU.DrawStringCentered("Put the hamsters on the buttons one
by one",CIwSVec2(SX*1,SY*0),SmallFont,BLACK,CIwSVec2(SX*3,SY*1));
        }
        else
        {
            O_MENU.DrawStringCentered("Check the + and -
First",CIwSVec2(SX*1,SY*0),SmallFont,BLACK,CIwSVec2(SX*3,SY*1));
        }
    }
    else
    TipScrolling("0");
    O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

    if(O_DIALOG.KeyEvent[0])
    {
        Dialog(1,1,5,4);
        O_MENU.DrawStringCenteredTop("Level 7: Complicated
BUTTON",CIwSVec2(SX*1,SY*1,2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
        O_TIPS.DrawTopLeftImage(1,CIwSVec2(SX*2.5,SY*2),125);
        O_MENU.DrawStringCentered("Play with the
button",CIwSVec2(SX*1,SY*3),SmallFont,BLACK,CIwSVec2(SX*6,SY*1));

        O_MENU.DrawStringCenteredTop("GOAL!",CIwSVec2(SX*1,SY*5),MediumFont,GREEN,CIwSV
ec2(SX*6,SY*1));
        O_MENU.DrawStringCentered("Light up all the
LEDs.",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));
    }

```

```

O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
        if(O_NEXT.PNR()==RELEASEINREGION)
        {
            O_DIALOG.KeyEvent[0] = false;
        }
    }
}

void Cstage1::Level8()
{
    /*
        L = FX
        0  1  2  3  4  5  6  7
    0  BK
    1
    2  FX0  -      ?0  -  L  -  -  -[
    3
    4
    5  B0
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_LED[0].Counter[0] = 3;

        /*Setting initial position*/

        O_LED[0].OriPos = CIwSVec2(SX*4,SY*2);
        O_BUTTON[0].OriPos = CIwSVec2(SX*0,SY*5);
        O_QUE[0].OriPos = CIwSVec2(SX*2,SY*2);
        O_DIALOG.KeyEvent[0]=true;
        O_BUTTON[0].KeyEvent[0]=true;
    }

    //WIRING
    WIRING(1,2,HOR,O_WIRE.KeyEvent[0]);
    //WIRING(0,2,HOR,O_WIRE.KeyEvent[0]);
    O_GROUND.DrawImage(0,CIwSVec2(0,SY*0),TOPLEFT);

    WIRING(3,2,HOR,O_WIRE.KeyEvent[0]);
    O_QUE[0].Position(CIwSVec2(SX*2,SY*2),TOPLEFT);

    WIRING(5,2,HOR,O_WIRE.KeyEvent[0]);
    WIRING(6,2,HOR,O_WIRE.KeyEvent[0]);
    O_LED[0].Position(CIwSVec2(SX*4,SY*2),TOPLEFT);
    //WIRING(7,2,HOR,O_WIRE.KeyEvent[0]);
    O_GROUND.DrawImage(1,CIwSVec2(SX*7,SY*0),TOPLEFT);

```

```

/*Interaction when touch*/
BUTTONDragging(0,0);

/*Puzzle Mechanism*/

if(O_BUTTON[0].Counter[0] == 0 && O_BUTTON[0].KeyEvent[0])
O_WIRE.KeyEvent[0]= true;
else
O_WIRE.KeyEvent[0]= false;

O_STAGE[7].KeyEvent[0] = O_WIRE.KeyEvent[0];

/*Draw components*/
QUEDrawing(0);

LEDDrawing(0,O_WIRE.KeyEvent[0]);
BUTTONDrawing(0,"A");

//Check PNR
Refreshing(1);

//TIPS Scrolling
O_MENU.DrawStringCentered("Drag the
Button!",CIwSVec2(SX*1,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));
O_MENU.DrawStringCentered("L =
A",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

if(O_DIALOG.KeyEvent[0])
{
Dialog(1,1,5,4);
O_MENU.DrawStringCenteredTop("Level 1: Gate
101",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
O_LED[0].DrawImage(1,CIwSVec2(SX*2.5,SY*2),TOPLEFT);
O_BUTTON[0].DrawImage(1,CIwSVec2(SX*4.5,SY*2),TOPLEFT);
O_ABC.DrawImage(0,CIwSVec2(SX*4.5,SY*2),TOPLEFT);
O_MENU.DrawStringCenteredTop("ON \nON
",CIwSVec2(SX*1,SY*3),MediumFont,GREEN,CIwSVec2(SX*6,SY*1));
O_MENU.DrawStringCenteredTop(" LED is L, \n buttons are A or
B or C.",CIwSVec2(SX*1,SY*3),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

O_MENU.DrawStringCentered("GOAL!",CIwSVec2(SX*1,SY*4),MediumFont,GREEN,CIwSVec2
(SX*6,SY*1));
O_MENU.DrawStringCentered("Satisfy this equation!\nL =
A",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
if(O_NEXT.PNR()==RELEASEINREGION)
{
O_DIALOG.KeyEvent[0] = false;
}
}
}

```

```

void Cstage1::Level9()
{
    /*
        L = FX
        0  1  2  3  4  5  6  7
    0  BK
    1
    2  FX0  -      ?0 ?1 L  -  -  -[
    3
    4
    5  B0
    */

    if (LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_LED[0].Counter[0] = 3;

        /*Setting initial position*/

        O_LED[0].OriPos = CIwSVec2(SX*4, SY*2);
        O_BUTTON[0].OriPos = CIwSVec2(SX*0, SY*5);
        O_BUTTON[1].OriPos = CIwSVec2(SX*1, SY*5);
        O_QUE[0].OriPos = CIwSVec2(SX*2, SY*2);
        O_QUE[1].OriPos = CIwSVec2(SX*3, SY*2);
        O_DIALOG.KeyEvent[0]=true;
        O_BUTTON[0].KeyEvent[0]=true;
        O_BUTTON[1].KeyEvent[0]=true;
    }

    //WIRING
    WIRING(1, 2, HOR, O_WIRE.KeyEvent[0]);
    //WIRING(0, 2, HOR, O_WIRE.KeyEvent[0]);
    O_GROUND.DrawImage(0, CIwSVec2(0, SY*0), TOPLEFT);

    //WIRING(3, 2, HOR, O_WIRE.KeyEvent[0]);
    O_QUE[0].Position(CIwSVec2(SX*2, SY*2), TOPLEFT);
    O_QUE[1].Position(CIwSVec2(SX*3, SY*2), TOPLEFT);

    WIRING(5, 2, HOR, O_WIRE.KeyEvent[0]);
    WIRING(6, 2, HOR, O_WIRE.KeyEvent[0]);
    O_LED[0].Position(CIwSVec2(SX*4, SY*2), TOPLEFT);
    //WIRING(7, 2, HOR, O_WIRE.KeyEvent[0]);
    O_GROUND.DrawImage(1, CIwSVec2(SX*7, SY*0), TOPLEFT);

    /*Interaction when touch*/
    BUTTONDragging(0, 1);
    BUTTONDragging(1, 1);
}

```

```
/*Puzzle Mechanism*/

if(
    O_QUE[0]. KeyEvent[0]&&
    O_QUE[1]. KeyEvent[0]&&
    O_BUTTON[0]. KeyEvent[0]&&
    O_BUTTON[1]. KeyEvent[0]
)
O_WIRE. KeyEvent[0]= true;
else
O_WIRE. KeyEvent[0]= false;

O_STAGE[8]. KeyEvent[0] = O_WIRE. KeyEvent[0];

/*Draw components*/
QUEDrawing(0);
QUEDrawing(1);

LEDDrawing(0, O_WIRE. KeyEvent[0]);
BUTTONDrawing(0, "A");
BUTTONDrawing(1, "B");

//Check PNR
Refreshing(1);

//TIPS Scrolling
O_MENU. DrawStringCentered("Drag the
Button!", CIwSVec2(SX*1, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));
O_MENU. DrawStringCentered("L = A and
B", CIwSVec2(SX*4, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));

if(O_DIALOG. KeyEvent[0])
{
    Dialog(1, 1, 5, 4);
    O_MENU. DrawStringCenteredTop("Level 2:
AND", CIwSVec2(SX*1, SY*1.2), MediumFont, BLACK, CIwSVec2(SX*6, SY*0.8));

    O_MENU. DrawStringCenteredTop("AND =
Series", CIwSVec2(SX*1, SY*2), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

    O_MENU. DrawStringCentered("GOAL!", CIwSVec2(SX*1, SY*4), MediumFont, GREEN, CIwSVec2
(SX*6, SY*1));
    O_MENU. DrawStringCentered("Satisfy this equation!\nL = A and
B", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

    O_NEXT. DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX
, SY));
    if(O_NEXT. PNR()==RELEASEINREGION)
    {
        O_DIALOG. KeyEvent[0] = false;
    }
}
}
void Cstage1::Level10()
```



```

{
    /*
        0  1  2  3  4  5  6  7
    0  BK
    1
    2  -      -      T  -  L  -  -  -[
    3
    4          CBL  ?0  -      -  -  -[
    5  B0
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

        O_LED[0].Counter[0] = 3;

        /*Setting initial position*/

        O_LED[0].OriPos = CIwSVec2(SX*4, SY*2);
        O_BUTTON[0].OriPos = CIwSVec2(SX*0, SY*5);
        O_QUE[0].OriPos = CIwSVec2(SX*3, SY*3);
        O_DIALOG.KeyEvent[0]=true;
        O_BUTTON[0].KeyEvent[0]=true;
    }

    //WIRING

    O_GROUND.DrawImage(0, CIwSVec2(0, SY*0), TOPLEFT);

    WIRING(2, 2, TT, true);
    WIRING(1, 2, HOR, true);
    WIRING(3, 2, HOR, O_WIRE.KeyEvent[0]);
    WIRING(2, 3, CBL, !O_WIRE.KeyEvent[0]);
    O_QUE[0].Position(CIwSVec2(SX*3, SY*3), TOPLEFT);
    WIRING(4, 3, HOR, !O_WIRE.KeyEvent[0]);
    WIRING(5, 3, HOR, !O_WIRE.KeyEvent[0]);
    WIRING(6, 3, HOR, !O_WIRE.KeyEvent[0]);

    O_LED[0].Position(CIwSVec2(SX*4, SY*2), TOPLEFT);
    WIRING(5, 2, HOR, O_WIRE.KeyEvent[0]);
    WIRING(6, 2, HOR, O_WIRE.KeyEvent[0]);
    O_GROUND.DrawImage(1, CIwSVec2(SX*7, SY*0), TOPLEFT);

    /*Interaction when touch*/
    BUTTONDragging(0, 1);
    BUTTONDragging(1, 1);

    /*Puzzle Mechanism*/

```

```

        if( O_QUE[0].KeyEvent[0] && O_BUTTON[0].KeyEvent[0])
        O_WIRE.KeyEvent[0]= false;
        else
        O_WIRE.KeyEvent[0]= true;

        O_STAGE[9].KeyEvent[0] = O_QUE[0].KeyEvent[0]
&& !O_BUTTON[0].KeyEvent[0];
        /*Draw components*/
        QUEDrawing(0);

        LEDDrawing(0,O_WIRE.KeyEvent[0]);
        BUTTONDrawing(0,"A");

        //Check PNR
        Refreshing(1);

        //TIPS Scrolling
        O_MENU.DrawStringCentered("Drag the
Button!",CIwSVec2(SX*1,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));
        O_MENU.DrawStringCentered("L =
not(A)",CIwSVec2(SX*4,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));

        if(O_DIALOG.KeyEvent[0])
        {
            Dialog(1,1,5,4);
            O_MENU.DrawStringCenteredTop("Level 3: Not
A",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));
            O_BUTTON[0].DrawImage(0,CIwSVec2(SX*3.5,SY*2),TOPLEFT);
            O_ABC.DrawImage(3,CIwSVec2(SX*3.5,SY*2),TOPLEFT);
            O_MENU.DrawStringCenteredTop("Not A = Button A in off
mode\nRemember, the goal is to satisfy the equation,\nNot to light up the
LED",CIwSVec2(SX*1,SY*3),SmallFont,BLACK,CIwSVec2(SX*6,SY*1));

            O_MENU.DrawStringCentered("GOAL!",CIwSVec2(SX*1,SY*4),MediumFont,GREEN,CIwSVec2
(SX*6,SY*1));
            O_MENU.DrawStringCentered("Satisfy this equation!\nL =
not(A)",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

            O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
            if(O_NEXT.PNR()==RELEASEINREGION)
            {
                O_DIALOG.KeyEvent[0] = false;
            }
        }
    }
    void Cstage1::Level11()
    {
        /*
            0 1 2 3 4 5 6 7
            0 BK
            1
        */
    }

```

```

2      -      -      T  ?0  T  - L  --[

3          CBL  ?1  CBR          [

4          [

5      B0  B1          [

*/

if (LEVELSTART)
{
    LEVELSTART = false;

    ClearData();

    /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

    O_LED[0].Counter[0] = 3;

    /*Setting initial position*/

    O_LED[0].OriPos = CIwSVec2(SX*6, SY*2);
    O_BUTTON[0].OriPos = CIwSVec2(SX*0, SY*5);
    O_BUTTON[1].OriPos = CIwSVec2(SX*1, SY*5);
    O_QUE[0].OriPos = CIwSVec2(SX*3, SY*2);
    O_QUE[1].OriPos = CIwSVec2(SX*3, SY*3);
    O_DIALOG.KeyEvent[0]=true;
    O_BUTTON[0].KeyEvent[0]=true;
    O_BUTTON[1].KeyEvent[0]=true;
}

//WIRING

O_GROUND.DrawImage(0, CIwSVec2(0, SY*0), TOPLEFT);

WIRING(1, 2, HOR, O_WIRE.KeyEvent[0]);
WIRING(2, 2, TT, O_WIRE.KeyEvent[0]);
WIRING(2, 3, CBL, O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0]);
O_QUE[0].Position(CIwSVec2(SX*3, SY*2), TOPLEFT);
O_QUE[1].Position(CIwSVec2(SX*3, SY*3), TOPLEFT);
WIRING(4, 2, TT, O_WIRE.KeyEvent[0]);
WIRING(4, 3, CBR, O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0]);

O_LED[0].Position(CIwSVec2(SX*5, SY*2), TOPLEFT);
WIRING(6, 2, HOR, O_WIRE.KeyEvent[0]);
O_GROUND.DrawImage(1, CIwSVec2(SX*7, SY*0), TOPLEFT);

/*Interaction when touch*/
BUTTONDragging(0, 1);
BUTTONDragging(1, 1);

/*Puzzle Mechanism*/

if( ( O_QUE[0].KeyEvent[0] &&
O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0] ) ||
( O_QUE[1].KeyEvent[0] && O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0] ) )

```

```
O_WIRE.KeyEvent[0]= true;
else
O_WIRE.KeyEvent[0]= false;

O_STAGE[10].KeyEvent[0] = O_QUE[0].KeyEvent[0] && O_QUE[1].KeyEvent[0]
&&
O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0] &&
O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0];

/*Draw components*/
QUEDrawing(0);
QUEDrawing(1);

LEDDrawing(0, O_WIRE.KeyEvent[0]);
BUTTONDrawing(0, "A");
BUTTONDrawing(1, "B");

//Check PNR
Refreshing(1);

//TIPS Scrolling
O_MENU.DrawStringCentered("Drag the
Button!", CIwSVec2(SX*1, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));
O_MENU.DrawStringCentered("L = A or
B", CIwSVec2(SX*4, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));

if(O_DIALOG.KeyEvent[0])
{
Dialog(1, 1, 5, 4);
O_MENU.DrawStringCenteredTop("Level 4:
OR", CIwSVec2(SX*1, SY*1. 2), MediumFont, BLACK, CIwSVec2(SX*6, SY*0. 8));

O_MENU.DrawStringCenteredTop("Or =
Parallel", CIwSVec2(SX*1, SY*2), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

O_MENU.DrawStringCentered("GOAL!", CIwSVec2(SX*1, SY*4), MediumFont, GREEN, CIwSVec2
(SX*6, SY*1));
O_MENU.DrawStringCentered("Satisfy this equation!\nL = A or
B", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

O_NEXT.DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX
, SY));
if(O_NEXT.PNR()==RELEASEINREGION)
{
O_DIALOG.KeyEvent[0] = false;
}
}
}
void Cstage1::Level12()
{
/*
0 1 2 3 4 5 6 7
```

```

0      BK                                     [
1                                     [
2      -      -      ?0 ?1 T L - [

3                                     CBL      ?2 - [
4                                     [
5      B0 B1 B2                                     [
*/

if(LEVELSTART)
{
    LEVELSTART = false;

    ClearData();

    /*Setting number of hamster, LED and ROLL initial plus minus
sign*/

    O_LED[0].Counter[0] = 3;

    /*Setting initial position*/

    O_LED[0].OriPos = CIwSVec2(SX*5, SY*2);
    O_BUTTON[0].OriPos = CIwSVec2(SX*0, SY*5);
    O_BUTTON[1].OriPos = CIwSVec2(SX*1, SY*5);
    O_BUTTON[2].OriPos = CIwSVec2(SX*2, SY*5);
    O_QUE[0].OriPos = CIwSVec2(SX*2, SY*2);
    O_QUE[1].OriPos = CIwSVec2(SX*3, SY*2);
    O_QUE[2].OriPos = CIwSVec2(SX*5, SY*3);
    O_DIALOG.KeyEvent[0]=true;
    O_BUTTON[0].KeyEvent[0]=true;
    O_BUTTON[1].KeyEvent[0]=true;
    O_BUTTON[2].KeyEvent[0]=true;
}

//WIRING

O_GROUND.DrawImage(0, CIwSVec2(0, SY*0), TOPLEFT);

WIRING(1, 2, HOR, O_WIRE.KeyEvent[0] || O_WIRE.KeyEvent[1]);
WIRING(4, 2, TT, O_WIRE.KeyEvent[0] || O_WIRE.KeyEvent[1]);
WIRING(4, 3, CBL, O_WIRE.KeyEvent[1]);
O_QUE[2].Position(CIwSVec2(SX*5, SY*3), TOPLEFT);
O_QUE[0].Position(CIwSVec2(SX*2, SY*2), TOPLEFT);
O_QUE[1].Position(CIwSVec2(SX*3, SY*2), TOPLEFT);

O_LED[0].Position(CIwSVec2(SX*5, SY*2), TOPLEFT);
WIRING(6, 2, HOR, O_WIRE.KeyEvent[0]);
WIRING(6, 3, HOR, O_WIRE.KeyEvent[1]);
O_GROUND.DrawImage(1, CIwSVec2(SX*7, SY*0), TOPLEFT);

/*Interaction when touch*/
BUTTONDragging(0, 2);
BUTTONDragging(1, 2);
BUTTONDragging(2, 2);

```

```

/*Puzzle Mechanism*/

if( ( O_QUE[0].KeyEvent[0] &&
O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0])&&
    ( O_QUE[1].KeyEvent[0] && O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0]))
{
    if(!O_QUE[2].KeyEvent[0])
    {
        O_WIRE.KeyEvent[0]= true;
        O_WIRE.KeyEvent[1]= false;
    }
    else
    {
        if(!O_BUTTON[O_QUE[2].Counter[0]].KeyEvent[0])
        {
            O_WIRE.KeyEvent[0]= true;
            O_WIRE.KeyEvent[1]= false;
        }
        else
        {
            O_WIRE.KeyEvent[0]= false;
            O_WIRE.KeyEvent[1]= true;
        }
    }
}
else
{
    O_WIRE.KeyEvent[0]= false;
    O_WIRE.KeyEvent[1]= false;
}

O_STAGE[11].KeyEvent[0] = O_QUE[0].KeyEvent[0] && O_QUE[1].KeyEvent[0]
&& O_QUE[2].KeyEvent[0] &&
( (O_BUTTON[0].Counter[0]
== 0 && O_BUTTON[1].Counter[0] == 1) ||
(O_BUTTON[0].Counter[0]
== 1 && O_BUTTON[1].Counter[0] == 0))&& O_BUTTON[2].Counter[0] == 2 &&
O_WIRE.KeyEvent[0];

/*Draw components*/
QUEDrawing(0);
QUEDrawing(1);
QUEDrawing(2);

LEDDrawing(0, O_WIRE.KeyEvent[0]);
BUTTONDrawing(0, "A");
BUTTONDrawing(1, "B");
BUTTONDrawing(2, "C");

//Check PNR
Refreshing(1);
O_MENU.DrawStringCentered("Drag the
Button!", CIwSVec2(SX*1, SY*0), MediumFont, BLACK, CIwSVec2(SX*3, SY*1));

```

```

        O_MENU.DrawStringCentered("L = A and B and
not(C)", CIwSVec2(SX*4, SY*0), SmallFont, BLACK, CIwSVec2(SX*3, SY*1));

        if(O_DIALOG.KeyEvent[0])
        {
            Dialog(1, 1, 5, 4);
            O_MENU.DrawStringCenteredTop("Level 5:
Combined!", CIwSVec2(SX*1, SY*1. 2), MediumFont, BLACK, CIwSVec2(SX*6, SY*0. 8));

            O_MENU.DrawStringCenteredTop("Combined", CIwSVec2(SX*1, SY*2), MediumFont, BLACK, CI
wSVec2(SX*6, SY*1));

            O_MENU.DrawStringCentered("GOAL!", CIwSVec2(SX*1, SY*4), MediumFont, GREEN, CIwSVec2
(SX*6, SY*1));

            O_MENU.DrawStringCentered("Satisfy this equation!\nL = A and B
and not(C)", CIwSVec2(SX*1, SY*5), MediumFont, BLACK, CIwSVec2(SX*6, SY*1));

            O_NEXT.DrawStringCentered("SKIP", CIwSVec2(SX*6, SY*5), MediumFont, RED, CIwSVec2(SX
, SY));

            if(O_NEXT.PNR()==RELEASEINREGION)
            {
                O_DIALOG.KeyEvent[0] = false;
            }
        }
}
void Cstagal::Level13()
{
    /*
        0   1   2   3   4   5   6   7
    0   BK                                     [
    1                                     CTL?0   CTR   [
    2   -   -   -   X ?1   X   -   -   [
    3                                     TL ?2   TR   -   -   [
    4                                     CBL L   CBR                                     [
    5   B0  B1  B2                                     [
    */

    if(LEVELSTART)
    {
        LEVELSTART = false;

        ClearData();

        /*Setting number of hamster, LED and ROLL initial plus minus
sign*/
        O_LED[0].Counter[0] = 3;

        /*Setting initial position*/

        O_LED[0].OriPos = CIwSVec2(SX*3, SY*4);
        O_BUTTON[0].OriPos = CIwSVec2(SX*0, SY*4);
        O_BUTTON[1].OriPos = CIwSVec2(SX*0, SY*5);
        O_BUTTON[2].OriPos = CIwSVec2(SX*6, SY*5);
    }
}

```

```
O_QUE[0].OriPos = CIwSVec2(SX*3, SY*1);
O_QUE[1].OriPos = CIwSVec2(SX*3, SY*2);
O_QUE[2].OriPos = CIwSVec2(SX*3, SY*3);
O_DIALOG.KeyEvent[0]=true;
O_BUTTON[0].KeyEvent[0]=true;
O_BUTTON[1].KeyEvent[0]=true;
O_BUTTON[2].KeyEvent[0]=true;
}

//WIRING

WIRING(1, 2, HOR, 1);
O_GROUND.DrawImage(0, CIwSVec2(0, SY*0), TOPLEFT);

WIRING(2, 1, CTL, O_WIRE.KeyEvent[1]);
WIRING(2, 2, X, 1);
WIRING(2, 3, TL, O_WIRE.KeyEvent[2] || O_WIRE.KeyEvent[0]);
WIRING(2, 4, CBL, O_WIRE.KeyEvent[0]);

WIRING(4, 1, CTR, O_WIRE.KeyEvent[1]);
WIRING(4, 2, X, 1);
WIRING(4, 3, TR, O_WIRE.KeyEvent[2] || O_WIRE.KeyEvent[0]);
WIRING(4, 4, CBR, O_WIRE.KeyEvent[0]);

O_QUE[0].Position(CIwSVec2(SX*3, SY*1), TOPLEFT);
O_QUE[1].Position(CIwSVec2(SX*3, SY*2), TOPLEFT);
O_QUE[2].Position(CIwSVec2(SX*3, SY*3), TOPLEFT);

O_LED[0].Position(CIwSVec2(SX*3, SY*4), TOPLEFT);

WIRING(5, 2, HOR, O_WIRE.KeyEvent[0] || O_WIRE.KeyEvent[1] || O_WIRE.KeyEvent[2]);

WIRING(6, 2, HOR, O_WIRE.KeyEvent[0] || O_WIRE.KeyEvent[1] || O_WIRE.KeyEvent[2]);
O_GROUND.DrawImage(1, CIwSVec2(SX*7, SY*0), TOPLEFT);

/*Interaction when touch*/
BUTTONDragging(0, 2);
BUTTONDragging(1, 2);
BUTTONDragging(2, 2);

/*Puzzle Mechanism*/

if( ( O_QUE[0].KeyEvent[0] && O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0])
    || ( O_QUE[1].KeyEvent[0] &&
O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0])
    || ( O_QUE[2].KeyEvent[0] &&
O_BUTTON[O_QUE[2].Counter[0]].KeyEvent[0]) )
{
    O_WIRE.KeyEvent[0]=false;
}
else
{
    O_WIRE.KeyEvent[0]= true;
}
}
```



```

        if( (O_QUE[0].KeyEvent[0] && O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0]))
            O_WIRE.KeyEvent[1]= true;
        else
            O_WIRE.KeyEvent[1]= false;

        if( (O_QUE[2].KeyEvent[0] && O_BUTTON[O_QUE[2].Counter[0]].KeyEvent[0]))
            O_WIRE.KeyEvent[2]= true;
        else
            O_WIRE.KeyEvent[2]= false;

        O_STAGE[12].KeyEvent[0] = (O_QUE[0].KeyEvent[0]
&& !O_BUTTON[O_QUE[0].Counter[0]].KeyEvent[0])&&
                                (O_QUE[1].KeyEvent[0]
&& !O_BUTTON[O_QUE[1].Counter[0]].KeyEvent[0])&&
                                (O_QUE[2].KeyEvent[0]
&& !O_BUTTON[O_QUE[2].Counter[0]].KeyEvent[0]);

        /*Draw components*/
        QUEDrawing(0);
        QUEDrawing(1);
        QUEDrawing(2);

        LEDDrawing(0,O_WIRE.KeyEvent[0]);
        BUTTONDrawing(0,"A");
        BUTTONDrawing(1,"B");
        BUTTONDrawing(2,"C");

        //Check PNR
        Refreshing(1);

        //TIPS Scrolling
        O_MENU.DrawStringCentered("Drag the
Button!",CIwSVec2(SX*1,SY*0),MediumFont,BLACK,CIwSVec2(SX*3,SY*1));
        O_MENU.DrawStringCentered("L = not(A)and not(B) \nand
not(C)",CIwSVec2(SX*4,SY*0),SmallFont,BLACK,CIwSVec2(SX*3,SY*1));

        if(O_DIALOG.KeyEvent[0])
        {
            Dialog(1,1,5,4);
            O_MENU.DrawStringCenteredTop("Level 6:
NoNoNo!",CIwSVec2(SX*1,SY*1.2),MediumFont,BLACK,CIwSVec2(SX*6,SY*0.8));

            O_MENU.DrawStringCenteredTop("It's useful to have this graphical
programing tool.",CIwSVec2(SX*1,SY*2),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

            O_MENU.DrawStringCentered("GOAL!",CIwSVec2(SX*1,SY*4),MediumFont,GREEN,CIwSVec2
(SX*6,SY*1));

            O_MENU.DrawStringCentered("Satisfy this equation!\nL = not(A)and
not(B) and not(C)",CIwSVec2(SX*1,SY*5),MediumFont,BLACK,CIwSVec2(SX*6,SY*1));

            O_NEXT.DrawStringCentered("SKIP",CIwSVec2(SX*6,SY*5),MediumFont,RED,CIwSVec2(SX
,SY));
    
```

```
        if(O_NEXT.PNR()==RELEASEINREGION)
        {
            O_DIALOG.KeyEvent[0] = false;
        }
    }
}
void Cstage1::Dialog(int16 TOPLX,int16 TOPLY,int16 WID,int16 HEI)
{
    Iw2DSetColour(0xffffffff) ;
    Iw2DFillRect(CIwSVec2(SX*TOPLX,SY*TOPLY),
CIwSVec2(SX*(WID+1),SY*(HEI+1)));

    O_FRAME2.DrawImage(0,CIwSVec2(SX*TOPLX,SY*TOPLY),TOPLEFT,IW_2D_IMAGE_TRANSFORM_NONE);

    O_FRAME2.DrawImage(0,CIwSVec2(SX*TOPLX,SY*(TOPLY+HEI)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_FLIP_Y);

    O_FRAME2.DrawImage(0,CIwSVec2(SX*(TOPLX+WID),SY*(TOPLY)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_FLIP_X);

    O_FRAME2.DrawImage(0,CIwSVec2(SX*(TOPLX+WID),SY*(TOPLY+HEI)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_ROT180);

    for(int i=TOPLY;i<=HEI;i++)
    {

        O_FRAME2.DrawImage(1,CIwSVec2(SX*TOPLX,SY*i),TOPLEFT,IW_2D_IMAGE_TRANSFORM_ROT270);

        O_FRAME2.DrawImage(2,CIwSVec2(SX*TOPLX,SY*(i+1)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_ROT270);

        O_FRAME2.DrawImage(1,CIwSVec2(SX*(TOPLX+WID),SY*(i+1)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_ROT90);

        O_FRAME2.DrawImage(2,CIwSVec2(SX*(TOPLX+WID),SY*i),TOPLEFT,IW_2D_IMAGE_TRANSFORM_ROT90);

    }

    for(int i=TOPLX;i<=WID;i++)
    {

        O_FRAME2.DrawImage(1,CIwSVec2(SX*(i+1),SY*TOPLY),TOPLEFT,IW_2D_IMAGE_TRANSFORM_NONE);

        O_FRAME2.DrawImage(2,CIwSVec2(SX*i,SY*TOPLY),TOPLEFT,IW_2D_IMAGE_TRANSFORM_NONE);

        O_FRAME2.DrawImage(1,CIwSVec2(SX*(i+1),SY*(TOPLY+HEI)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_FLIP_Y);

        O_FRAME2.DrawImage(2,CIwSVec2(SX*i,SY*(TOPLY+HEI)),TOPLEFT,IW_2D_IMAGE_TRANSFORM_FLIP_Y);

    }
}
```

```

}
void Cstage1::ClearData()
{
    O_DIALOG.KeyEvent[0]=false;
    for(int j=0;j<5;j++)
    {
        O_WIRE.KeyEvent[j];
        for(int i=0;i<4;i++)
        {
            O_ROLL[i].KeyEvent[j] = false;
            O_ROLL[i].Counter[j] = 0;
            O_LED[i].KeyEvent[j] = false;
            O_LED[i].Counter[j] = 0;
            O_BUTTON[i].KeyEvent[j] = false;
            O_BUTTON[i].Counter[j] = 100;
            O_QUE[i].KeyEvent[j] = false;
            O_QUE[i].Counter[j] = 100;
        }
    }

    for(int i=0;i<4;i++)
    {
        O_ROLL[i].TouchMech = UNTOUCH;
        O_LED[i].TouchMech = UNTOUCH;
        O_BUTTON[i].TouchMech = UNTOUCH;
    }
    O_HB.TouchMech = UNTOUCH;
    O_BACK.TouchMech = UNTOUCH;

    O_NEXT.TouchMech = UNTOUCH;
    O_RESET.TouchMech = UNTOUCH;

    //TIPS
    TipsTimer = s3eTimerGetMs();
    ScrollTimer = s3eTimerGetMs();
    TIPS = 0;
    Scroll190 = 100;

    //Default First tips
    O_TIPS.Counter[0] = 0;
    O_TIPS.Counter[1] = 0;
}
void Cstage1::Update()
{
    s3eDeviceYield(0);
    s3eKeyboardUpdate();
    s3ePointerUpdate();
    Iw2DSurfaceShow();
}
void Cstage1::Refreshing(int16 ItemNumber)
{
    O_BACK.PNR();
    if(O_BACK.TouchMech == RELEASEINREGION)
    {
        LEVEL = 0;
        O_BACK.Counter[0] = 2;
    }
}

```

```

        LEVELSTART = true;
    }

    if(O_RESET.TouchMech == RELEASEINREGION)
    {
        LEVELSTART = true;
    }

    if(O_STAGE[LEVEL-1].KeyEvent[0] && !O_DIALOG.KeyEvent[0])
    {
        O_NEXT.DrawImage(0, CIwSVec2(SX*7, SY*0), TOPLEFT);
        O_NEXT.PNR();
        if(O_NEXT.TouchMech == RELEASEINREGION)
        {
            LEVELSTART = true;
            LEVEL++;
            if(LEVEL==8)
            {
                LEVEL = 0;
                O_BACK.Counter[0] = 2;
            }
        }
    }
    else if(!O_DIALOG.KeyEvent[0])
    {
        O_RESET.PNR();
        O_RESET.DrawImage(0, CIwSVec2(SX*7, 0), TOPLEFT);
    }

    O_TIPSFRAME.DrawImage(0, CIwSVec2(SX*1, 0), TOPLEFT);
    O_GOALFRAME.DrawImage(0, CIwSVec2(SX*4, 0), TOPLEFT);
    O_BACK.DrawImage(0, CIwSVec2(0, 0), TOPLEFT);
}

void Cstage1::ROLLInteraction(int16 i)
{
    O_ROLL[i].PNR(O_ROLL[i].OriPos, O_ROLL[i].Size);
    if(O_ROLL[i].TouchMech == RELEASENOTINREGION && O_ROLL[i].KeyEvent[0])
    {
        O_HB.Counter[0]++;
        O_ROLL[i].KeyEvent[0] = false;
    }
    else if(O_ROLL[i].TouchMech == RELEASEINREGION)
    {
        O_ROLL[i].Counter[0] = (O_ROLL[i].Counter[0] + 1)%4;
    }
}

void Cstage1::ROLLCollision(int16 i)
{
    if(O_HB.TouchMech == RELEASENOTINREGION)
    {
        if(Collision(O_HB.CPos, O_ROLL[i].CPos) && O_HB.Counter[0] > 0
&& !O_ROLL[i].KeyEvent[0])
        {
            O_HB.Counter[0]--;

```

```
        O_ROLL[i].KeyEvent[0] = true; //TRIGGER
    }
}
void Cstage1::ROLLDrawing(int16 i)
{
    if(!O_ROLL[i].KeyEvent[0])
        O_ROLL[i].DrawImage(0);
    else
    {
        O_ROLL[i].DrawAnimation("1232",100);
    }

    switch(O_ROLL[i].Counter[0])
    {
        case 0: //+ // -
            {
                O_PlusMinus.DrawImage(0,O_ROLL[i].Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_NONE);
                break;
            }
        case 1: // - +
            {
                O_PlusMinus.DrawImage(1,O_ROLL[i].Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_FLIP_X);
                break;
            }
        case 2: // - // +
            {
                O_PlusMinus.DrawImage(0,O_ROLL[i].Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_FLIP_Y);
                break;
            }
        case 3: // + -
            {
                O_PlusMinus.DrawImage(1,O_ROLL[i].Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_NONE);
                break;
            }
    }
}
void Cstage1::BUTTONInteraction(int16 i)
{
    O_BUTTON[i].PNR(O_BUTTON[i].OriPos, O_BUTTON[i].Size);
    if(O_BUTTON[i].TouchMech == RELEASENOTINREGION && O_BUTTON[i].KeyEvent[0])
    {
        O_HF.Counter[0]++;
        O_BUTTON[i].KeyEvent[0] = false;
    }
}
void Cstage1::BUTTONCollision(int16 i)
```

```

{
    if(O_HF.TouchMech == RELEASENOTINREGION)
    {
        if(Collision(O_HF.CPos,O_BUTTON[i].CPos) && O_HF.Counter[0] > 0
&& !O_BUTTON[i].KeyEvent[0] )
        {
            O_HF.Counter[0]--;
            O_BUTTON[i].KeyEvent[0] = true;        //TRIGGER
        }
    }
}
void Cstage1::BUTTONDragging(int16 i,int16 QueI)
{
    if(!O_DIALOG.KeyEvent[0])
    {
        if(O_BUTTON[i].KeyEvent[2])
        {
            O_BUTTON[i].PNR(O_QUE[O_BUTTON[i].Counter[0]].OriPos,O_QUE[i].Size);
        }
        else
        {
            O_BUTTON[i].PNR(O_BUTTON[i].OriPos,O_BUTTON[i].Size);
        }

        if(O_BUTTON[i].KeyEvent[2])
        {
            switch (O_BUTTON[i].TouchMech)
            {
                case HOLDNOTINREGION:
                {
                    O_BUTTON[i].Position(CIwSVec2(ReturnSingleTouch()->x,ReturnSingleTouch()-
>y),CENTER);

                    O_BUTTON[i].KeyEvent[1] = true;
                    break;
                }
                case RELEASENOTINREGION:
                {
                    O_BUTTON[i].KeyEvent[2] = false;
                    O_QUE[O_BUTTON[i].Counter[0]].KeyEvent[0]
= false;

                    for(int k=0;k<=QueI;k++)
                    {
                        if(Collision(O_BUTTON[i].CPos,O_QUE[k].CPos) && !O_QUE[k].KeyEvent[0] )
                        {
                            O_QUE[O_BUTTON[i].Counter[0]].KeyEvent[0] = false;

                            O_BUTTON[i].Position(O_QUE[k].Topleft,TOPLEFT);

                            O_BUTTON[i].KeyEvent[1] =
true;

                            O_BUTTON[i].KeyEvent[2] =
true;

```

```

true; //TRIGGER

O_QUE[k]. KeyEvent[0] =
O_BUTTON[i]. Counter[0]=k;
O_QUE[k]. Counter[0]=i;
break;
}
}

if(!O_BUTTON[i]. KeyEvent[2])
{
O_QUE[O_BUTTON[i]. Counter[0]]. KeyEvent[0] = false;
O_BUTTON[i]. KeyEvent[1] = false;
O_BUTTON[i]. Counter[0]=100;

O_BUTTON[i]. Position(O_BUTTON[i]. OriPos, TOPLEFT);
}

break;
}
case RELEASEINREGION:
{

O_BUTTON[i]. Position(O_QUE[O_BUTTON[i]. Counter[0]]. Topleft, TOPLEFT);
O_BUTTON[i]. KeyEvent[0]
= !O_BUTTON[i]. KeyEvent[0];

break;
}
}
else
{
switch (O_BUTTON[i]. TouchMech)
{
case HOLDNOTINREGION:
{

O_BUTTON[i]. Position(CIwSVec2(ReturnSingleTouch()->x, ReturnSingleTouch()-
>y), CENTER);

O_BUTTON[i]. KeyEvent[1] = true;
break;
}
case HOLDINREGION:
{
if(O_BUTTON[i]. KeyEvent[1])
{

O_BUTTON[i]. Position(CIwSVec2(ReturnSingleTouch()->x, ReturnSingleTouch()-
>y), CENTER);
}
break;
}
case RELEASENOTINREGION:
{
for(int k=0;k<=QueI;k++)
{

```

```

        if(Collision(O_BUTTON[i].CPos,O_QUE[k].CPos) && !O_QUE[k].KeyEvent[0] )
            {

                O_BUTTON[i].Position(O_QUE[k].Topleft, TOPLEFT);

                O_BUTTON[i].KeyEvent[1] =
true;                                O_BUTTON[i].KeyEvent[2] =
true;                                O_QUE[k].KeyEvent[0] =
true; //TRIGGER                    O_BUTTON[i].Counter[0]=k;
                                    O_QUE[k].Counter[0]=i;
                                    break;
            }
        }

        if(!O_BUTTON[i].KeyEvent[2])
        {
            O_BUTTON[i].Counter[0]=100;
            O_BUTTON[i].KeyEvent[1] = false;

            O_BUTTON[i].Position(O_BUTTON[i].OriPos, TOPLEFT);
            }
            break;
        }
        case RELEASEINREGION:
            {

                O_BUTTON[i].Position(O_BUTTON[i].OriPos, TOPLEFT);
                if(O_BUTTON[i].KeyEvent[1])
                {
                    O_BUTTON[i].KeyEvent[1]=false;
                }
                else
                {
                    O_BUTTON[i].KeyEvent[0]
=O_BUTTON[i].KeyEvent[0];
                }
            }
            break;
        }
    }
}

void Cstage1::QUEDrawing(int16 i)
{
    O_QUE[i].DrawImage(0,O_QUE[i].OriPos, TOPLEFT);
}
void Cstage1::BUTTONDrawing(int16 i)
{
    if(!O_BUTTON[i].KeyEvent[0])
    O_BUTTON[i].DrawImage(0);
}

```



```

else
{
    if(O_BUTTON[i].KeyEvent[0])
    {
        O_BUTTON[i].DrawAnimation("1232",100);
    }
    else
        O_BUTTON[i].DrawImage(0,O_HF.OriPos, TOPLEFT);
}
}
void Cstage1::BUTTONDrawing(int16 i,const char* String)
{
    if(!O_BUTTON[i].KeyEvent[0])
    {
        O_BUTTON[i].DrawImage(0);
    }
    else
    {
        if(O_BUTTON[i].KeyEvent[1])
        {
            O_BUTTON[i].DrawAnimation("1232",100);
        }
        else
            O_BUTTON[i].DrawAnimation("1232",100,O_BUTTON[i].OriPos);
    }
    if(String == "A")
    {
        O_ABC.DrawImage(0+3*!O_BUTTON[i].KeyEvent[0],O_BUTTON[i].Topleft, TOPLEFT);
    }
    else if(String == "B")
    {
        O_ABC.DrawImage(1+3*!O_BUTTON[i].KeyEvent[0],O_BUTTON[i].Topleft, TOPLEFT);
    }
    else if(String == "C")
    {
        O_ABC.DrawImage(2+3*!O_BUTTON[i].KeyEvent[0],O_BUTTON[i].Topleft, TOPLEFT);
    }
    else
    {
        O_ABC.DrawImage(2+3*!O_BUTTON[i].KeyEvent[0],O_BUTTON[i].Topleft, TOPLEFT);
    }
}
void Cstage1::LEDInteraction(int16 i)
{
    O_LED[i].PNR(O_LED[i].OriPos,O_LED[i].Size);
    if(O_LED[i].TouchMech == RELEASEINREGION )
    {
        O_LED[i].Counter[0] = (O_LED[i].Counter[0] + 1)%4;
    }
}
void Cstage1::LEDDrawing(int16 i, bool KeyEVENT)
{

```

```

O_LED[i]. DrawImage(KeyEVENT);
switch(O_LED[i]. Counter[0])
{
    case 0: //+
        // -
        {
            O_PlusMinus. DrawImage(0, O_LED[i]. Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_NONE);
            break;
        }
    case 1: // - +
        {
            O_PlusMinus. DrawImage(1, O_LED[i]. Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_FLIP_X);
            break;
        }
    case 2: // -
        // +
        {
            O_PlusMinus. DrawImage(0, O_LED[i]. Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_FLIP_Y);
            break;
        }
    case 3: // + -
        {
            O_PlusMinus. DrawImage(1, O_LED[i]. Topleft, TOPLEFT, IW_2D_IMAGE_TRANSFORM_NONE);
            break;
        }
    }
}
}
void Cstapel::HBDrawing()
{
    if(O_HB. KeyEvent[0])
    {
        O_HB. DrawImage(0);
    }
    else
        O_HB. DrawImage(0, O_HB. OriPos, TOPLEFT);

    O_INFONUMBER. DrawImage(O_HB. Counter[0], O_HB. Topleft, TOPLEFT);
}
void Cstapel::HFDrawing()
{
    if(O_HF. KeyEvent[0])
    {
        O_HF. DrawImage(0);
    }
    else
        O_HF. DrawImage(0, O_HF. OriPos, TOPLEFT);

    O_INFONUMBER. DrawImage(O_HF. Counter[0], O_HF. Topleft, TOPLEFT);
}
}
void Cstapel::HBInteraction()
{

```

```

        if(!O_DIALOG.KeyEvent[0])
        O_HB.PNR(O_HB.OriPos, O_HB.Size);
        switch (O_HB.TouchMech)
        {
        case HOLDNOTINREGION:
            if(O_HB.Counter[0] > 0)
            {
                O_HB.Position(CIwSVec2(ReturnSingleTouch()-
>x, ReturnSingleTouch()->y), CENTER);
                O_HB.KeyEvent[0] = true;
                break;
            }
        default:
            O_HB.KeyEvent[0] = false;
            break;
        }
    }
}
void Cstage1::HFInteraction()
{
    if(!O_DIALOG.KeyEvent[0])
    O_HF.PNR(O_HF.OriPos, O_HF.Size);
    switch (O_HF.TouchMech)
    {
    case HOLDNOTINREGION:
        if(O_HF.Counter[0] > 0)
        {
            O_HF.Position(CIwSVec2(ReturnSingleTouch()-
>x, ReturnSingleTouch()->y), CENTER);
            O_HF.KeyEvent[0] = true;
            break;
        }
    default:
        O_HF.KeyEvent[0] = false;
        break;
    }
}
}
void Cstage1::TipScrolling(char const* FrameSequence)
{
    if(s3eTimerGetMs() > ScrollTimer + 10)
    {
        ScrollTimer = s3eTimerGetMs();
        Scroll190 = Scroll190+50;
        if(Scroll190>100)
        Scroll190 = 100;
    }

    if(s3eTimerGetMs() > TipsTimer + 5000 || O_TIPFRAME.TouchMech ==
PRESSRELEASEINREGION || TEMPFRAME!=*FrameSequence)
    {
        O_TIPS.KeyEvent[0] = false;
        TipsTimer = s3eTimerGetMs();
        Scroll190 = 0;
        if( *(FrameSequence + TIPS+1) !=NULL)
        {
            TIPS++;
        }
    }
}

```

```

        else
        {
            TIPS = 0;
        }
        O_TIPS.Counter[0] = O_TIPS.Counter[1];
        O_TIPS.Counter[1] = *(FrameSequence+TIPS)-48;
    }
    else if(O_TIPSFRAME.TouchMech == HOLDINREGION)
    {
        TipsTimer = s3eTimerGetMs();
    }

    if(O_TIPS.Counter[0] != O_TIPS.Counter[1])
    {
        if(O_TIPS.Counter[1]<8)
        {

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[0],O_TIPSFRAME.Topleft,CIwSVec2(0,Scro
1190),CIwSVec2(0,0));

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[1],O_TIPSFRAME.Topleft,CIwSVec2(0,0),C
IwSVec2(0,100-Scro1190));
        }
        else
        {

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[0],O_TIPSFRAME.Topleft,CIwSVec2(0,Scro
1190),CIwSVec2(0,0));

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[1],O_TIPSFRAME.Topleft,CIwSVec2(0,0),C
IwSVec2(0,100-Scro1190));
        }
    }
    else
    {
        if(O_TIPS.Counter[1]<8)
        {

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[0],O_TIPSFRAME.Topleft,CIwSVec2(0,0),C
IwSVec2(0,0));
        }
        else
        {

            O_TIPS.DrawScrollingImage(O_TIPS.Counter[0],O_TIPSFRAME.Topleft,CIwSVec2(0,0),C
IwSVec2(0,0));
        }
    }
    TEMPFRAME = *FrameSequence;
}
void Cstagal::WIRING(int16 PosX, int16 PosY, WIREMODE SHAPE,int16 VOLTAGE)
{
    int TEMP;
    TEMP=VOLTAGE*4;

    if(SHAPE == HOR)

```

```
{  
  
    O_WIRE.DrawImage(0+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_NONE);  
    }  
    else if(SHAPE == CTL)  
    {  
  
        O_WIRE.DrawImage(1+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT180);  
        }  
        else if(SHAPE == CTR)  
        {  
  
            O_WIRE.DrawImage(1+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT270);  
            }  
            else if(SHAPE == CBL)  
            {  
  
                O_WIRE.DrawImage(1+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT90);  
                }  
                else if(SHAPE == CBR)  
                {  
  
                    O_WIRE.DrawImage(1+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_NONE);  
                    }  
                    else if(SHAPE == TT)  
                    {  
  
                        O_WIRE.DrawImage(3+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT270);  
                        }  
                        else if(SHAPE == TR)  
                        {  
  
                            O_WIRE.DrawImage(3+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_NONE);  
                            }  
                            else if(SHAPE == TL)  
                            {  
  
                                O_WIRE.DrawImage(3+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT180);  
                                }  
                                else if(SHAPE == TB)  
                                {  
  
                                    O_WIRE.DrawImage(3+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM  
_ROT90);  
                                    }  
                                    else if(SHAPE == X)  
                                    {
```

```
O_WIRE.DrawImage(2+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM
_NONE);
    }
    else if(SHAPE == VER)
    {

O_WIRE.DrawImage(0+TEMP, CIwSVec2(PosX*SX, PosY*SY), TOPLEFT, IW_2D_IMAGE_TRANSFORM
_ROT90);
    }

}
bool Cstage1::Collision(CIwSVec2 OBJ1CPos, CIwSVec2 OBJ2CPos)
{
    if(( OBJ1CPos.x - OBJ2CPos.x < SX/4 && OBJ1CPos.x - OBJ2CPos.x >0 ) ||
(OBJ2CPos.x - OBJ1CPos.x < SX/4 && OBJ2CPos.x - OBJ1CPos.x >0) )
        if((OBJ1CPos.y - OBJ2CPos.y < SX/4 && OBJ1CPos.y - OBJ2CPos.y >0 ) ||
(OBJ2CPos.y - OBJ1CPos.y < SX/4 && OBJ2CPos.y - OBJ1CPos.y >0))
            return true;

    return false;
}
void Cstage1::SoundInit()
{
s3eAudioIsCodecSupported(S3E_AUDIO_CODEC_MP3);
s3eAudioRegister(S3E_AUDIO_STOP, 0, NULL);
}
```