STI ACADEMIC CENTER – CALAMBA

RINDO RUNNER

A Project Proposal Presented to
STI Academic Center – Calamba

In Partial Fulfillment
Of the Requirement for
Software Engineering

by:

Caballero, Nicolle R.
Magpantay, John Emmanuel R.
Roque, Marc Christian
Samiano, Dominique I.

Domenand Yedra
Adviser

STI ACADEMIC CENTER – CALAMBA

PANEL'S APPROVAL SHEET

This Project Proposal entitled

RINDO RUNNER

Developed by:

Caballero, Nicolle R.
Magpantay, John Emmanuel R.
Roque, Marc Christian
Samiano, Dominique I.

After having been presented is hereby approved
by the following members of the panel

_____

Panelist

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Background of the Study

A platform game is a video game which involves guiding an avatar to jump between suspended platforms, over obstacles, or both to advance the game. These challenges are known as jumping puzzles or free- running. The player controls the jumps to avoid letting the avatar fall from platforms or miss necessary jumps. The most common unifying element of games of this genre is the jump button. Jumping, in this genre, may include swinging from extendable arms, as in Ristar or Bionic Commando, or bouncing from springboards or trampolines, as in Alpha Waves. These mechanics, even in the context of other genres, are commonly called platforming. Games where jumping is automated completely, such as The Legend of Zelda, fall outside of the genre. Platform games originated in the early 1980s. At one point, platform games were the most popular genre of video game. At the peak of their popularity, it is estimated that between one-quarter and one-third of console games were platformers. No genre

either before or since has been able to achieve a similar market share. As of 2006, the genre had become far less dominant, representing a two percent market share as compared to fifteen percent in 1998, but is still commercially viable, with a number of games selling in the millions of units. Since 2010, a variety of endless running platformers for mobile devices have brought renewed popularity to the genre.

Dinosaurs are a group of reptiles with a set of physical features that are different from those of all other reptiles. They include extinct animals we know from fossils and the birds we see today. The extinct animals we normally think of as dinosaurs had their heyday in the Mesozoic. The word 'dinosaur' means 'terrible lizard' in Greek. It was coined in 1842 by Sir Richard Owen, an English Professor of Comparative Anatomy and Physiology. Dinosaurs lived between 230 and 65 million years ago, in a time known as the Mesozoic Era. This was many millions of years before the first modern humans, Homo sapiens, appeared. Scientists divide the Mesozoic Era into 3 periods: the Triassic, Jurassic and Cretaceous. The Cretaceous-Tertiary extinction event, or the K-T event, is the

name given to the die-off of the dinosaurs and other species. For many years, palaeontologists believed this event was caused by climate and geological changes that interrupted the dinosaurs' food supply. However, in the 1980s, father-and-son scientists Luis (1911-88) and Walter Alvarez (1940-) discovered in the geological record a distinct layer of iridium–an element found in abundance only in space–that corresponds to the precise time the dinosaurs died. This suggests that a comet, asteroid or meteor impact event may have caused the extinction of the dinosaurs. In the 1990s, scientists located the massive Chicxulub Crater at the tip of Mexico's Yucatán Peninsula, which dates to the period in question.

1.2 Statement of the Problem

1.2.1 General Problem

How will the player going to enjoy and be entertained playing the game while learning the information about few Dinosaurs, their types, families and characteristics?

1.2.2 Specific Problem

Lot of children gain information or knowledge from watching a movie,

Short story from books and games that is why many of them this days was ill-informed or unfamiliar about the Prehistoric period. How we will bring back the attention of them regarding Dinosaurs and How to provide interactive game about Prehistoric Creatures?

1.3 Objectives of the Study

1.3.1 General Objectives

The Proponents decided to create a 2D platform game called Rindo Runner.

1.3.2 Specific Objective

The Proponents will provide information about Dinosaurs using platform game and we will create an adventure and strategic game with storyboard or trivia to provide interaction with the player that will revitalize their perception about Prehistoric Creatures.

1.4 Significance of the Study

- The proponents plan to introduce the game in exciting way.
- We made this system to retrieve the attention of people back to the prehistoric time.
- The player can gain different lessons from playing this game, one of this is Perseverance, and like the other games this take a lot of time to complete this game.
- Moreover, when the user fail a level, they often return to the last checkpoint.
- This teaches users/player to be patient and keep trying until they succeed.

1.5 Scope and Limitations

1.5.1 Scope

- A free to play game that doesn't require account creation.
- It can be played as a Story mode game.
- It can be played without internet connection.
- It will provide information about Prehistoric Creatures.
- It can be played as single player mode.
- It will show attractive arts and storylines.
- It can be played with different stages.

1.5.2 Limitations

- The game can only be played using computer, it will not be able to play on android phones or any other game gadgets like Xbox.
- Other than mouse and keyboard, the game will not support other peripherals like joystick and gamepads.
- The game only have first stage and second stage.
- You cannot play it multiplayer.
- It does not display the full information about dinosaurs.

- The story of Rindo Hubert is incomplete.

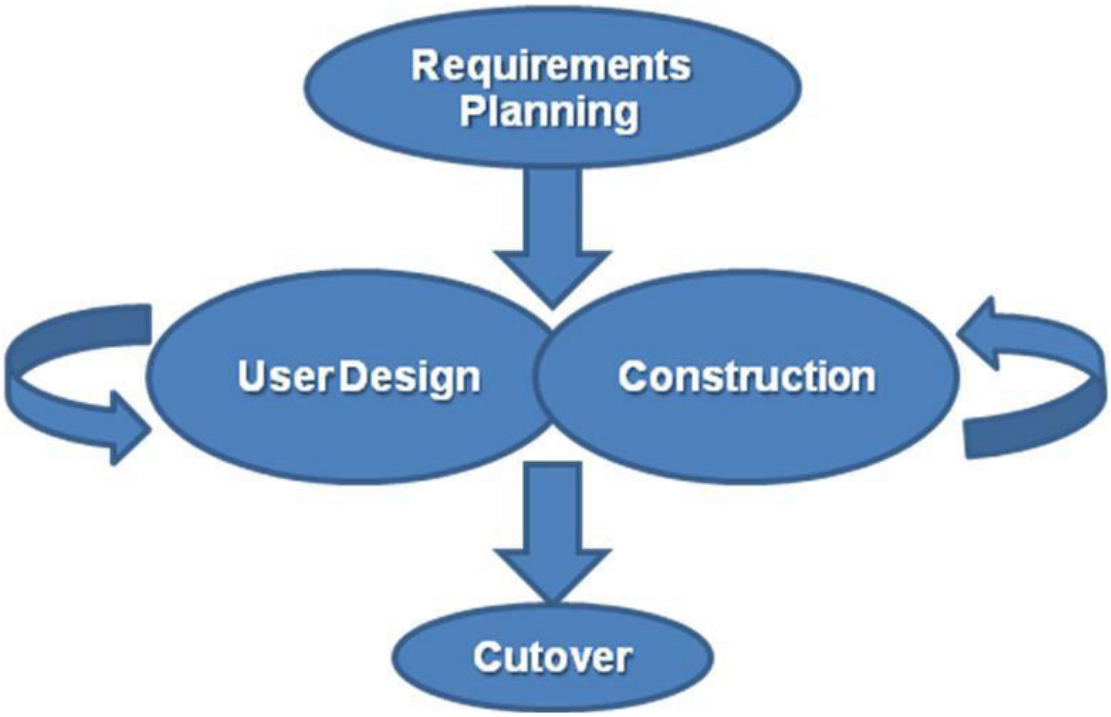
Chapter 2

METHODOLOGY OF THE STUDY

2.1 RAD Methodology

.

RAD methodology is used for planning, developing, designing, and implementation the game software. **Rapid application development** is both a general term used to refer to alternatives to the conventional waterfall model of software development.



Requirements planning phase – In this phase, the proponents look for software and other reference that will be needed to create a game that will suffice to display and provide the information about the topic.

User design phase – The proponents use reference of sprites and background from the internet. The game design is based on typical 2D endless running game.

Construction phase – In this phase, the proponents add more commands and movements in the system like, jumping and double jump, crystal collection and reward system.

Cutover phase – In this phase, the testing and bug fixing are made. The game will be sooner delivered on the users.

Chapter 3

DATA GATHERING PROCEDURES AND OUTPUTS

3.1 Data Gathering Instrument

The proponents do a survey and took 100 respondents from different places like Makiling National High school and people here in Calamba.

SURVEY

Name: _____ Age: _____

Grade Level: _____ Address: _____

School: _____ _____

Part 1. A-questions (Awareness)

A1. Have you heard/seen about dinosaurs before?

- ☐ **Yes**
- ☐ **No (Proceed to question A5)**

A2. If Yes, where did you seen/heard them? (Select as many as you can)

- ☐ **Magazine**
- ☐ **Books**
- ☐ **Teachers**
- ☐ **Movie**
- ☐ **Internet**
- ☐ **Social Media**
- ☐ **Other (Please specify) _____**

A3. If A1=yes, Do you find the resources helpful?

- ☐ **Yes**
- ☐ **No**

A4. Which dinosaurs you already know?

- ☐ **Tyrannosaurus Rex**
- ☐ **Sauropods**
- ☐ **Ceratopsians**
- ☐ **Raptors**
- ☐ **Stegosaurus**
- ☐ **Prosauropods**
- ☐ **Stegosaurs**
- ☐ **Ankylosaurs**
- ☐ **Hardosaurs**
- ☐ **Ornithomimids**
- ☐ **Ornithopods**
- ☐ **Others: _____**

A5. By any chance, would you like to know more about dinosaurs?

- ☐ **Yes**
- ☐ **No**

Part 2. B – Question (Learning)

B1. During the year of your studies, have you encountered Dinosaurs as a topic in your class?

- ☐ **Yes**

☐ No

B2. Are you interested about them?

☐ Yes
☐ No

B3. Do you think it is important to your studies to know more about dinosaurs? Rate your answer. From 1 – 5,

☐ 1 – Strongly disagree
☐ 2 – Disagree
☐ 3 – Neutral
☐ 4 – Agree
☐ 5 – Strongly Agree

Part 3. C – Question (Computer-based question)

C1. Does your school have a computer?

☐ Yes
☐ No

C2. If C1 = YES, How many?

☐ 1-10
☐ 11-20
☐ 21 – 30
☐ 31+

C3. Are you a computer user?

☐ Yes
☐ No

C4. Are you playing computer games?

☐ Yes
☐ No

C5. By any chance, would you like to play a computer game about dinosaurs?

☐ Yes
☐ No

C6. How would you like it to be? Please explain your answer as detailed as possible.

Thank you for answering our survey!

End of document

Figure 3.1 Survey Form

Survey Results

A1: Yes = 97%
No = 3%

A2: Magazine = 42
Books = 55
Teachers = 44
Movies = 77
Internet = 67
Social Media = 47

A3: Yes = 78%
No = 13%

A4: Tyrannosaurus Rex = 75
Sauropods = 18
Ceratopsians = 18
Raptors = 19
Stegosaurus = 28
Prosauropods = 14
Stegosaurs = 21
Ankylosaurs = 12

Hardosaurs = 15
Ormithomimids = 9
Ornithopods = 11

A5: Yes = 79%
No = 14%

B1: Yes = 63%
No = 31 %

B2: Yes = 76%
No = 19%

B3: 1- Strongly Disagree = 9
2- Disagree = 7
3- Neutral = 32
4- Agree = 38
5- Strongly Agree = 8

C1: Yes = 92%
No = 2%

C2: 1-10 = 34
11- 20 = 10
21 – 30 = 9
31 + = 40

C3 Yes = 98%
No = 2%

C4 Yes = 72%
No = 28%

C5 Yes = 81%
No = 19%

3.2 Sources of Data

- Brackeys Unity 3D Tutorial

  A video that provides the viewers a short tutorial on how to use a specific application. The proponents also do research for the improvement of the system.

- Library

  The thesis of the previous graduates at the library served as a reference for this research.

- Internet

  Source of the definitions of PC games and other terms used in this research.

Chapter 4

DOCUMENTATION OF THE CURRENT SYSTEM

4.1 Current System

A lot of movies was released with genres of Adventure, Thriller, Fantasy and Science Fiction, a good example for this matter was the Jurassic Park/World. It has a great description and amazing story about Prehistoric Creatures.

There are also a lot of movies featuring dinosaurs since the year of 1960's until now. The famous gigantic monster called Godzilla is one of them.

Dinosaurs also introduced with short story books and encyclopaedias with its history, others make a short game just to apprise the time where the prehistoric man exists.

Chapter 5

THE PROPOSED SYSTEM

5.1 System Overview

The game will give short storyboard regarding the story of the main character which you will use in game. A main menu will show which you can choose if ever you wanted to view the storyboard or proceed to new game already. You will select which level you wanted to play. Each level will show a dialog box which will provide an instruction for the player. A trivia and details about prehistoric creatures will be introduced after playing each stage of the game. A gallery of different dinosaurs will show and you can choose any from the 9 dinosaurs each stage you complete.

This game can provide knowledge and awareness about this creatures to the users who will play.

5.2 Sequence Diagram

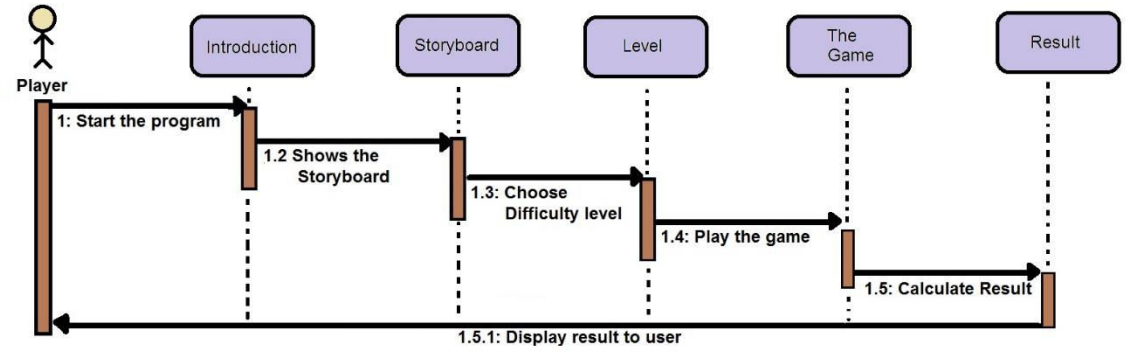


Figure 5.1 Sequence Diagram

5.3 Use Case Diagram

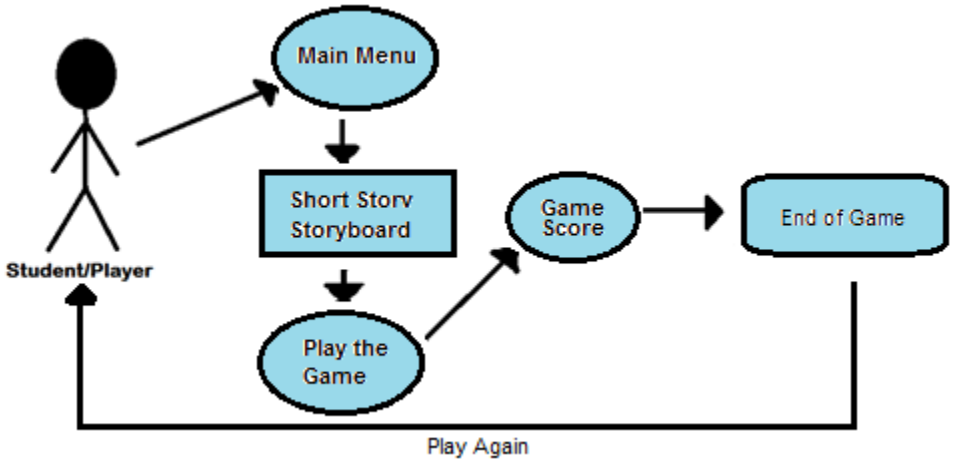


Figure 5.2 Use Case Diagram

5.4 Screen Layout

Figure 5.3 Main Menu

Figure 5.4 Storyboard

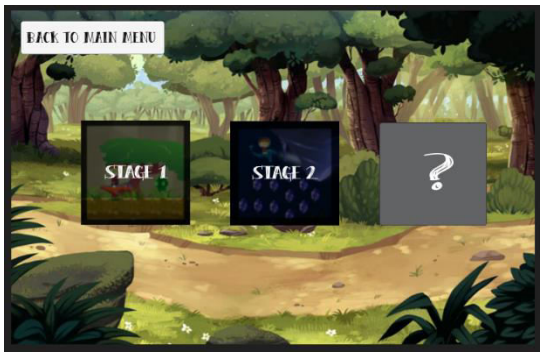


Figure 5.5 Stage Selection

Figure 5.6 Stage 1

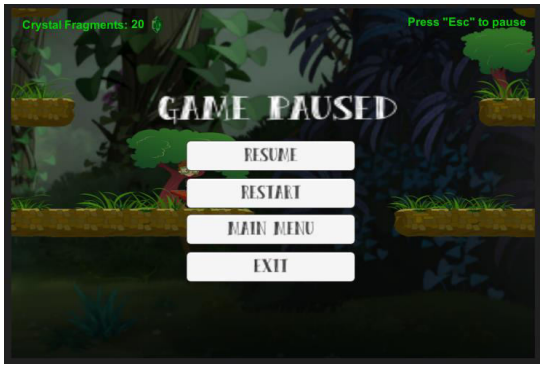


Figure 5.7 Paused Menu

Figure 5.8 Stage 2

Figure 5.9 Stage 1 Gallery

Figure 5.10 Stage 2 Gallery

Figure 5.11 Gallery 1 Sample

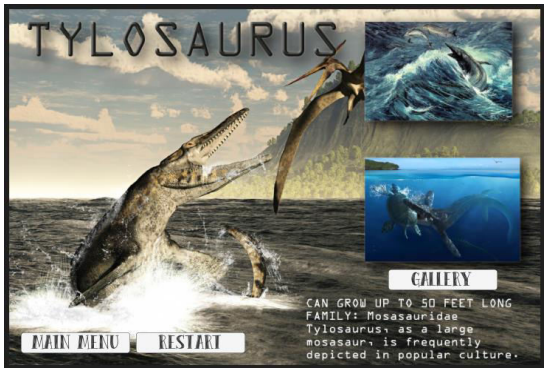


Figure 5.12 Gallery 2 Sample

Figure 5.13 Game Over

CHAPTER 6

SYSTEM REQUIREMENT SPECIFICATION

6.1 System Requirements Specification

6.1.1 Hardware Requirements

RAM	**1 GB**
Processor	**Dual Core or Higher**
Motherboard	**Pentium 4 2.4 GHz or Higher**
Hard Drive	**500gigabytes or Higher**
Video Card	**DirectX 9-capable(1024 x 768 or higher display resolution)**

6.1.2 Software Requirements

- Operating System (Windows XP SP2+, 7 SP1+, 8, 10; Mac OS X 10.8+)

Unity 5.0.0p2 (64-bit)	This is for running the game
QuickTime Player	Uploading videos to unity
BFXR	Playing the sound effects

6.1.3 Human Resource Requirements

- Users/Players
- Programmers

Chapter 7

SYSTEM MAINTENANCE PLAN

7.1 System Maintenance Plan

- Monthly check and update of the system

```
using UnityEngine;
using System.Collections;

public class ActivateTextAtLine : MonoBehaviour {

	public TextAsset theText;

	public int startLine;
	public int endLine;

	public TextBoxManager theTextBox;

	public bool destroyWhenActivated;

	public bool stopPlayer;

	void Start () {

		theTextBox = FindObjectOfType<TextBoxManager>();

	}

	void Update () {
	}

	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindo")
		{
			theTextBox.ReloadScript(theText);
			theTextBox.currentLine = startLine;
			theTextBox.endAtLine = endLine;
			theTextBox.EnableTextBox();

			if (destroyWhenActivated)
			{
				Destroy (gameObject);
			}
		}
	}


}

using UnityEngine;
using System.Collections;
```

```
public class ActivateTextAtLine : MonoBehaviour {

	public TextAsset theText;

	public int startLine;
	public int endLine;

	public TextBoxManager theTextBox;

	public bool destroyWhenActivated;

	public bool stopPlayer;

	void Start () {

		theTextBox = FindObjectOfType<TextBoxManager>();

	}

	void Update () {
	}

	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindo")
		{
			theTextBox.ReloadScript(theText);
			theTextBox.currentLine = startLine;
			theTextBox.endAtLine = endLine;
			theTextBox.EnableTextBox();

			if (destroyWhenActivated)
			{
				Destroy (gameObject);
			}
		}
	}


}
	using UnityEngine;
using System.Collections;

public class ChangeScene : MonoBehaviour {

	public void ChangeToScene (string sceneToChangeTo) {
		Application.LoadLevel (sceneToChangeTo);
```

```
	}
}

using UnityEngine;
using System.Collections;

public class Checkpoint : MonoBehaviour {
	public LevelManager levelManager;


	void Start () {
		levelManager = FindObjectOfType<LevelManager> ();

	}

	void Update () {

	}
	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindo") {
			levelManager.currentCheckpoint = gameObject;
		}
	}
}



using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class CrystalCollection : MonoBehaviour {

	public static int score;


	Text text;

	void Start()
	{
		text = GetComponent<Text>();

		score = 0;
	}

	void Update()
	{
```

```
		if (score < 0)
			score = 0;
		text.text = "" + score;
	}

	public static void AddCrystal (int crystalsToAdd)
	{
		score += crystalsToAdd;
	}
	public static void Reset()
	{
		score = 0;
	}
	void OnDisable(){
		PlayerPrefs.SetInt ("TotalScore", (int)score);
	}
}



using UnityEngine;
using System.Collections;

public class CrystalPicker : MonoBehaviour {

	public int crystalsToAdd;

	public AudioSource crystalPickup;

	void OnTriggerEnter2D (Collider2D other) {

		if (other.GetComponent<PlayerControll> () == null)
		return;
		CrystalCollection.AddCrystal (crystalsToAdd);
		crystalPickup.Play();
		Destroy (gameObject);


	}



}



using UnityEngine;
```

```
using System.Collections;

public class CrystalPickerSEA : MonoBehaviour {

	public int crystalsToAdd;

	public AudioSource crystalPickup;

	void OnTriggerEnter2D (Collider2D other) {

		if (other.GetComponent<PlayerControllSEA> () == null)
			return;
		CrystalCollection.AddCrystal (crystalsToAdd);
		crystalPickup.Play();
		Destroy (gameObject);


	}



}using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class GameOverpoints : MonoBehaviour {

	Text text;
	int score = 0;
	void Start () {
		score = PlayerPrefs.GetInt ("TotalScore");
		text = GetComponent<Text>();
		text.text = "" + score;
	}
	void update(){

	}

}

using UnityEngine;
using System.Collections;

public class InstructionPanel : MonoBehaviour {

	public GameObject InstUI;
	private bool show = false;
	void Start (){
```

```
		InstUI.SetActive (false);
	}
	void Update(){
		if(Input.GetButtonDown("show")){
			show = !show;

		}
		if(show){
			InstUI.SetActive(true);
			Time.timeScale=0;
		}

		if(!show){
			InstUI.SetActive(false);
			Time.timeScale=1;
		}
	}
}


using UnityEngine;
using System.Collections;

public class KillRindo : MonoBehaviour {

	public LevelManager levelManager;

	void Start () {
		levelManager = FindObjectOfType<LevelManager> ();
	}


	void Update () {

	}
	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindo") {
			Application.LoadLevel(3);
			GetComponent<AudioSource>().Play();
			return;
		}
	}
}


using UnityEngine;
```

```
using System.Collections;

public class KillRindoswimmer : MonoBehaviour {

	public LevelManager levelManager;

	void Start () {
		levelManager = FindObjectOfType<LevelManager> ();
	}


	void Update () {

	}
	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindoswimming") {
			Application.LoadLevel(3);
			GetComponent<AudioSource>().Play();
			return;
		}
	}
}


using UnityEngine;
using System.Collections;

public class LevelManager : MonoBehaviour {

	public GameObject currentCheckpoint;
	private PlayerControll player;
	public int crystalDeathPenalty;


	void Start () {
		player = FindObjectOfType<PlayerControll> ();
	}


	void Update () {

	}
	public void RespawnPlayer()
	{
		CrystalCollection.AddCrystal (-crystalDeathPenalty);
		Debug.Log ("Player Respawn");
```

```
            player.transform.position = currentCheckpoint.transform.position;

        }

}

#pragma strict
var music: AudioSource;

function QuitGame () {
        Debug.Log("Exiting Game...");
        Application.Quit ();
}

function SetGameVolume (vol: float) {
        music.volume = vol;
}


using UnityEngine;
using System.Collections;

public class PauseInterface : MonoBehaviour {

        public GameObject PauseUI;
        private bool paused = false;
        void Start (){
                PauseUI.SetActive (false);
}
        void Update(){
                if(Input.GetButtonDown("Pause")){
                        paused = !paused;

}
                if(paused){
                        PauseUI.SetActive(true);
                        Time.timeScale=0;
                }

                if(!paused){
                        PauseUI.SetActive(false);
                        Time.timeScale=1;
                }
        }
        public void Resume(){
                paused = false;
        }
```

```
	public void Restart(){
		Application.LoadLevel (Application.loadedLevel);
	}
	public void MainMenu(){
		Application.LoadLevel (0);
	}
	public void Quit(){
		Application.Quit ();
	}
}

using UnityEngine;
using System.Collections;

public class PlatformDestroyer : MonoBehaviour {

	public GameObject platformDestructionPoint;

	void Start () {
		platformDestructionPoint = GameObject.Find("PlatformDestructionPoint");
	}


	void Update () {


		if (transform.position.x < platformDestructionPoint.transform.position.x) {

			Destroy(gameObject);
		}
	}
}

using UnityEngine;
using System.Collections;

public class PlatformGenerator : MonoBehaviour {

	public GameObject thePlatform;
	public Transform generationPoint;
	public float distancebetween;

	public float distancebetweenMin;
	public float distancebetweenMax;

	private float platformWidth;
```

```
	void Start () {
		platformWidth = thePlatform.GetComponent<BoxCollider2D>().size.x;

	}


	void Update () {

		distancebetween = Random.Range (distancebetweenMin,
distancebetweenMax);
		if(transform.position.x < generationPoint.position.x)
		{
			transform.position = new Vector3(transform.position.x + platformWidth
+ distancebetween, transform.position.y, transform.position.z);
			Instantiate (thePlatform, transform.position, transform.rotation);
		}
	}
}


using UnityEngine;
using System.Collections;

public class PlayerControll : MonoBehaviour {
	//rindo takbo
	public float moveSpeed;
	public float jumpForce;
	private Rigidbody2D Rigidd;

	// talon rindo
	public bool grounded;
	public LayerMask whatIsGround;
	private Collider2D colliderr;
	private bool doubleJump;

	public bool canMove;

	// rindo animation
	private Animator animaTorr;

	//hp ni rindo
	public int currentHP;
	public int maxHP = 100;

	void Start () {
		Rigidd = GetComponent<Rigidbody2D> ();
		colliderr = GetComponent<Collider2D> ();
```

```
		animaTorr = GetComponent<Animator> ();

		currentHP = maxHP;
	}


	void Update () {

		if (!canMove){

				return;
			}


		grounded = Physics2D.IsTouchingLayers (colliderr, whatIsGround);

		Rigidd.velocity = new Vector2 (moveSpeed, Rigidd.velocity.y);
		// || Input.GetMouseButtonDown(0)
		if(Input.GetKeyDown(KeyCode.Space)){
			GetComponent<AudioSource>().Play();
			if(grounded){
			Rigidd.velocity = new Vector2(Rigidd.velocity.x, jumpForce);
			doubleJump = false;
			}
			if(!doubleJump && !grounded){
			Rigidd.velocity = new Vector2(Rigidd.velocity.x, jumpForce);
			doubleJump= true;
			}

			}

		animaTorr.SetFloat ("Speed", Rigidd.velocity.x);
		animaTorr.SetBool ("Ground", grounded);

	}
}


using UnityEngine;
using System.Collections;

public class PlayerControllSEA : MonoBehaviour {
	//rindo takbo
	public float moveSpeed;
	public float jumpForce;
	private Rigidbody2D Rigidd;
```

```
	// talon rindo

	private Collider2D colliderr;
	private bool doubleJump;

	// rindo animation
	private Animator animaTorr;


	void Start () {
		Rigidd = GetComponent<Rigidbody2D> ();
		colliderr = GetComponent<Collider2D> ();
		animaTorr = GetComponent<Animator> ();


	}


	void Update () {



		Rigidd.velocity = new Vector2 (moveSpeed, Rigidd.velocity.y);
		// || Input.GetMouseButtonDown(0)
		if(Input.GetKeyDown(KeyCode.Space)){
			GetComponent<AudioSource>().Play();
			Rigidd.velocity = new Vector2(Rigidd.velocity.x, jumpForce);
			}

		animaTorr.SetFloat ("Speed", Rigidd.velocity.x);


	}
}

using UnityEngine;
using System.Collections;

public class Scroll : MonoBehaviour {

	public float speed = 0;


	void Update () {
		Vector2 offset = new Vector2 (Time.time* speed, 0f);
		GetComponent<Renderer>().material.mainTextureOffset = offset;
	}
```

```
}

using UnityEngine;
using System.Collections;

public class Stage1toStage2 : MonoBehaviour {

	public LevelManager levelManager;


	void Update () {

	}
	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindo") {
			Application.LoadLevel(4);
			GetComponent<AudioSource>().Play();
			return;
		}
	}
}


using UnityEngine;
using System.Collections;

public class Stage2toStage3 : MonoBehaviour {

	public LevelManager levelManager;


	void Update () {

	}
	void OnTriggerEnter2D(Collider2D other)
	{
		if (other.name == "Rindoswimming") {
			Application.LoadLevel(6);
			GetComponent<AudioSource>().Play();
			return;
		}
	}
}

using UnityEngine;
using System.Collections;
```

```
using UnityEngine.UI;
public class TextBoxManager : MonoBehaviour {

	public GameObject textBox;

	public Text theText;


	public TextAsset textFile;
	public string[] textLines;

	public int currentLine;
	public int endAtLine;

	public PlayerControll player;

	public bool isActive;

	public bool stopPlayerMovement;

	private bool isTyping = false;
	private bool cancelTyping = false;

	public float typeSpeed;


	void Start () {

		player = FindObjectOfType<PlayerControll> ();

		if(textFile !=null)
		{
			textLines=(textFile.text.Split('\n'));
		}
		if (endAtLine == 0) {

			endAtLine = textLines.Length - 1;
		}
	}

	void Update()
	{
		//theText.text = textLines [currentLine];

		if (Input.GetKeyDown (KeyCode.Return))
		{
			if (!isTyping)
```

```
                {

                        currentLine += 1;

                        if (currentLine > endAtLine) {

                                DisableTextBox();
                        }
                        else
                        {
                                StartCoroutine(TextScroll(textLines[currentLine]));
                        }
                }
                else if(isTyping && !cancelTyping)
                {
                        cancelTyping = true;
                }
        }

}

private IEnumerator TextScroll (string lineOfText)
{
        int letter = 0;
        theText.text = "";
        isTyping = true;
        cancelTyping = false;
        while (isTyping && !cancelTyping && (letter < lineOfText.Length - 1))
        {
                theText.text += lineOfText[letter];
                letter += 1;
                yield return new WaitForSeconds(typeSpeed);

        }
        theText.text = lineOfText;
        isTyping = false;
        cancelTyping = false;
}

public void EnableTextBox()
{
        textBox.SetActive (true);
        isActive = true;


        if (stopPlayerMovement) {
```

```
                player.canMove = false;
            }

            StartCoroutine(TextScroll(textLines[currentLine]));
        }

        public void DisableTextBox()
        {
            textBox.SetActive (false);
            isActive = false;


            player.canMove = true;
        }

        public void ReloadScript(TextAsset theText)
        {
            if (theText != null) {

                textLines = new string[1];
                textLines=(theText.text.Split('\n'));
            }
        }
}
```