ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----- 🙡 🕮 🙣 -----



**PROJECT 3**

***Đề tài:* Thiết kế và lập trình game 2D giải đố.**

Giảng viên hướng dẫn: PGS.TS. Ngô Quỳnh Thu

Sinh viên thực hiện:

Họ tên: Mạch Quang Thiên

MSSV: 20215646

|  |
| --- |
|  |
|  |
|  |  |  |
|  |  |  |
|  |  |  |

**Hà Nội, tháng 12 năm 2024**

Contents

[LỜI NÓI ĐẦU 3](#_Toc186825624)

[CHƯƠNG I. MÔ TẢ BÀI TOÁN 4](#_Toc186825625)

[1. Mô tả yêu cầu bài toán 4](#_Toc186825626)

[CHƯƠNG II. CHỨC NĂNG CHÍNH, KỊCH BẢN HOẠT ĐỘNG CỦA NGƯỜI CHƠI 5](#_Toc186825627)

[1. Các chức năng chính 5](#_Toc186825628)

[2. Mô hình Use Case: 5](#_Toc186825641)

[3. Mô hình Class: 7](#_Toc186825641)

[4. Mô hình Tuần tự: 8](#_Toc186825641)

[5. Xây dựng chi tiết các class: 10](#_Toc186825641)

[CHƯƠNG III. CÁC CÔNG NGHỆ VÀ TÀI NGUYÊN SỬ DỤNG 22](#_Toc186825676)

[1. Xây dựng và triển khai game bằng Unity: 22](#_Toc186825677)

[2. Vẽ biểu đồ UML bằng StarUML: 22](#_Toc186825681)

[3. Sử dụng các asset có sẵn trên itch.io: 23](#_Toc186825685)

[CHƯƠNG IV: THIẾT KẾ VÀ TRIỂN KHAI SẢN PHẨM 24](#_Toc186825687)

[1. Thiết kế sản phẩm: 24](#_Toc186825688)

[a. Giao diện Intro: 24](#_Toc186825689)

[b. Giao diện Menu: 24](#_Toc186825691)

[c. Giao diện Gameplay: 25](#_Toc186825693)

[d. Giao diện Change Level: 25](#_Toc186825695)

[e. Giao diện Complete Game: 26](#_Toc186825697)

[2. Triển khai sản phẩm: 26](#_Toc186825699)

[CHƯƠNG V: TỔNG KẾT 15](#_Toc186825687)

[CHƯƠNG VI: TÀI LIỆU THAM KHẢO VÀ LIÊN KẾT 16](#_Toc186825687)

**LỜI NÓI ĐẦU**

Trò chơi điện tử không chỉ là một phương tiện giải trí mà còn mang lại cơ hội phát triển kỹ năng tư duy logic và sáng tạo. Từ niềm đam mê với lập trình và mong muốn tạo ra một sản phẩm giải trí hữu ích, em đã quyết định thực hiện dự án game này. Lý do chính để phát triển trò chơi là nhằm thử thách bản thân trong việc áp dụng kiến thức lập trình và thiết kế phần mềm vào thực tế, đồng thời mang lại trải nghiệm học tập thú vị qua một tựa game nhẹ nhàng và sáng tạo.

Mục tiêu của dự án là tạo ra một trò chơi mang tính giáo dục, thử thách người chơi với các câu hỏi toán học, rèn luyện kỹ năng tư duy và phản xạ. Qua quá trình phát triển game, em cũng mong muốn nâng cao kỹ năng lập trình, quản lý dự án, và thiết kế giao diện người dùng, từ đó hoàn thiện hơn khả năng làm việc trong lĩnh vực phát triển phần mềm.

Báo cáo này bao gồm các nội dung chính như sau:

* **Chương 1**: Mô tả bài toán.
* **Chương 2**: Chức năng chính, kịch bản hoạt động của người chơi.
* **Chương 3**: Các công nghệ và tài nguyên sử dụng.
* **Chương 4**: Thiết kế và triển khai sản phẩm.
* **Chương 5**: Tổng kết.
* **Chương 6:** Tài liệu tham khảo và liên kết.

Cuối cùng, em xin gửi lời cảm ơn chân thành đến giáo viên hướng dẫn Ngô Quỳnh Thu, người đã tận tình chỉ bảo và hỗ trợ em trong suốt quá trình thực hiện dự án. Những góp ý và định hướng quý báu từ cô đã giúp em vượt qua những khó khăn và hoàn thành trò chơi này một cách tốt nhất có thể.

Hy vọng rằng dự án này sẽ là một bước đệm quan trọng trong hành trình học tập và phát triển bản thân của em.

**CHƯƠNG I. MÔ TẢ BÀI TOÁN**

1. **Mô tả yêu cầu bài toán**

Trong thời đại công nghệ số, trò chơi không chỉ mang tính giải trí mà còn có thể trở thành công cụ giáo dục hiệu quả. Đặc biệt, các trò chơi liên quan đến toán học và logic không chỉ giúp người chơi thư giãn mà còn phát triển tư duy, khả năng giải quyết vấn đề và kỹ năng tính toán.

Đề tài "Quiz Game" được chọn với mong muốn tạo ra một sản phẩm vừa học vừa chơi, giúp người dùng rèn luyện trí tuệ một cách vui vẻ, không áp lực. Đây cũng là cơ hội để người phát triển rèn luyện kỹ năng lập trình và làm quen với các công cụ phát triển phần mềm hiện đại như Unity và ngôn ngữ C#.

Các yêu cầu cơ bản của hệ thống:

* Về chức năng:
  + Hệ thống được quản lý linh hoạt, có phân chia cấp độ.
  + Giao diện người dùng dễ nhìn, đơn giản, tập trung vào trải nghiệm giải toán.
  + Phản hồi nhanh chóng khi người chơi nhập đáp án.
* Về kỹ thuật:
  + Sử dụng Unity, hỗ trợ đa nền tảng.
  + Áp dụng ngôn ngữ C# để viết logic trò chơi.
  + Có khả năng lưu trữ tiến trình người chơi.
* Về nội dung:
  + Cung cấp 30 cấp độ với các câu hỏi toán học ngắn gọn, phù hợp mọi đối tượng.
  + Giao diện tối giản, sử dụng các gam màu đơn giản để người chơi tập trung vào nội dung trò chơi.

**CHƯƠNG II. CHỨC NĂNG CHÍNH, KỊCH BẢN HOẠT ĐỘNG CỦA NGƯỜI CHƠI**

1. **Các chức năng chính**

* **Chức năng chính của trò chơi:**
  + Hệ thống cấp độ: Trò chơi gồm nhiều cấp độ với độ khó tăng dần.
  + Hiển thị nội dung câu hỏi: Trò chơi hiển thị nội dung câu hỏi để người chơi có thể đọc.
  + Người chơi nhập kết quả: Người chơi sẽ nhập câu trả lời thông qua bàn phím ảo hoặc bàn phím thiết bị.
  + Xác nhận kết quả: Hệ thống kiểm tra và trả về thông báo đúng/sai ngay lập tức.
  + Quản lý tiến trình: Lưu trữ cấp độ hiện tại, đánh dấu các cấp độ đã hoàn thành.
  + Giao diện người dùng: Giao diện thân thiện, sử dụng các yếu tố cơ bản để tập trung vào trải nghiệm trò chơi.
* **Kịch bản hoạt động người chơi:**
  + Bắt đầu chơi: Người chơi bấm nút “Play” từ giao diện menu chính.
  + Giải câu đố: Nhập đáp án và xác nhận bằng nút “Enter”. Nếu đúng, chuyển sang cấp độ tiếp theo, nếu sai, thông báo cho người chơi thử lại.
  + Hoàn thành trò chơi: Khi hoàn thành các cấp độ, người chơi nhận thông báo chiến thắng.
  + Tạm dừng và lưu trò chơi: Khi người chơi muốn tạm dừng, người chơi sẽ bấm vào biểu tượng “<” để quay lại menu chính, hệ thống sẽ tự động lưu lại tiến trình của người chơi.

1. **Mô hình Use Case:**
   1. **Danh sách các tác nhân:**

|  |  |  |
| --- | --- | --- |
| STT | Tên tác nhân | Chức năng |
| 1 | Người chơi | * Bắt đầu chơi * Giải câu đố * Hoàn thành trò chơi * Tạm dừng và lưu trò chơi |

* 1. **Danh sách các use case:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Mã use case | Tên use case | Tác nhân tương tác |
| 1 | UC01 | Bắt đầu chơi | Người chơi |
| 2 | UC02 | Giải câu đố | Người chơi |
| 3 | UC03 | Hoàn thành trò chơi | Người chơi |
| 4 | UC04 | Tạm dừng và lưu trò chơi | Người chơi |

* 1. **Biểu đồ Use Case tổng quan:**

**A diagram of a diagram

Description automatically generated**

1. **Mô hình class:**
   1. **Danh sách các class:**

|  |  |  |
| --- | --- | --- |
| STT | Tên | Chức năng |
| 1 | QuestionDisplay | Hiển thị nội dung câu hỏi. |
| 2 | QuestionGenerate | Đọc câu hỏi từ file question.txt, phân giải cấu trúc câu hỏi. |
| 3 | AnswerGenerate | Đọc câu trả lời từ phía người chơi, gửi câu trả lời về hệ thống. |
| 4 | CanvasManager | Quản lý các canvas hiển thị của game. |
| 5 | LevelManager | Quản lý cấp độ của game, lưu tiến trình của người chơi. |
| 6 | ButtonManager | Quản lý các nút bấm có trong game. |

* 1. **Biểu đồ class tổng quan:**

**A diagram of a computer

Description automatically generated**

1. **Mô hình tuần tự:**
   1. **Danh sách các kịch bản:**

|  |  |  |
| --- | --- | --- |
| STT | Tên | Chức năng |
| 1 | Bắt đầu chơi | Mô tả kịch bản quy trình bắt đầu chơi |
| 2 | Giải câu đố | Mô tả kịch bản quy trình giải câu đố |
| 3 | Tạm dừng và lưu trò chơi | Mô tả kịch bản quy trình tạm dừng và lưu trò chơi |

* 1. **Biểu đồ tuần tự:**
     1. **Biểu đồ tuần tự quy trình “Bắt đầu chơi”:**

**A diagram of a company

Description automatically generated**

* + 1. **Biểu đồ tuần tự quy trình “Giải câu đố”:**

A diagram of a table

Description automatically generated with medium confidence

* + 1. **Biểu đồ tuần tự quy trình “Tạm dừng và lưu trò chơi”:**

**A diagram of a project

Description automatically generated**

1. **Xây dựng chi tiết các class:**
   1. **AnswerGenerate.cs:**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class AnswerGenerate : MonoBehaviour

{

public static string answer;

public GameObject screenAnswer;

public GameObject wrongText;

public AudioSource correctFX;

public AudioSource wrongFX;

public AudioSource congratulationFX;

void Start()

{

wrongText.SetActive(false);

}

public void EnterButton()

{

answer = screenAnswer.GetComponent<TMP\_InputField>().text;

screenAnswer.GetComponent<TMP\_InputField>().text = "";

if (QuestionGenerate.actualAnswer != answer)

{

wrongText.SetActive(true);

wrongFX.Play();

StartCoroutine(HideWrongTextAfterDelay(1f));

}

else if (QuestionGenerate.actualAnswer == answer)

{

//correctFX.Play();

if (wrongText.activeSelf)

{

wrongText.SetActive(false);

}

if (LevelManager.Instance.CurrentLevel == LevelManager.Instance.NumLevel)

{

congratulationFX.Play();

LevelManager.Instance.ResetBestLevel();

CanvasManager.Instance.ShowCompleteGameCanvas();

}

else

{

correctFX.Play();

CanvasManager.Instance.ShowChangeLevelCanvas();

LevelManager.Instance.CurrentLevel++;

LevelManager.Instance.UpdateBestLevel();

}

}

}

public void ChangeLevelButton()

{

CanvasManager.Instance.ShowGameplayCanvas();

QuestionGenerate.displayQuestion = false;

}

private IEnumerator HideWrongTextAfterDelay(float delay)

{

yield return new WaitForSeconds(delay);

wrongText.SetActive(false);

}

public void Button\_1()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "1";

}

public void Button\_2()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "2";

}

public void Button\_3()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "3";

}

public void Button\_4()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "4";

}

public void Button\_5()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "5";

}

public void Button\_6()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "6";

}

public void Button\_7()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "7";

}

public void Button\_8()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "8";

}

public void Button\_9()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "9";

}

public void Button\_0()

{

screenAnswer.GetComponent<TMP\_InputField>().text += "0";

}

}

* 1. **ButtonManager.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ButtonManager : MonoBehaviour

{

public void PlayButton()

{

LevelManager.Instance.CurrentLevel = LevelManager.Instance.BestLevel;

QuestionGenerate.displayQuestion = false;

CanvasManager.Instance.ShowGameplayCanvas();

}

public void BackButton()

{

CanvasManager.Instance.ShowMenuCanvas();

}

}

* 1. **CanvasManager.cs:**

using System.Collections;

using UnityEngine;

public class CanvasManager : MonoBehaviour

{

public static CanvasManager Instance; // Singleton

public GameObject menuCanvas;

public GameObject gameplayCanvas;

public GameObject changeLevelCanvas;

public GameObject levelStageCanvas;

public GameObject introCanvas;

public GameObject completeGameCanvas;

void Awake()

{

// Đảm bảo chỉ có một instance

if (Instance == null)

{

Instance = this;

DontDestroyOnLoad(gameObject);

}

else

{

Destroy(gameObject);

}

}

void Start()

{

StartCoroutine(ShowMenuAfterIntro());

}

private IEnumerator ShowMenuAfterIntro()

{

ShowIntroCanvas();

yield return new WaitForSeconds(4f);

ShowMenuCanvas();

}

public void ShowIntroCanvas()

{

introCanvas.SetActive(true);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(false);

}

public void ShowMenuCanvas()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(true);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(false);

}

public void ShowGameplayCanvas()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(true);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(false);

}

public void ShowChangeLevelCanvas()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(true);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(false);

}

public void ShowLevelStageCanvas()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(true);

completeGameCanvas.SetActive(false);

}

public void ShowCompleteGameCanvas()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(true);

}

public void HideAllCanvases()

{

introCanvas.SetActive(false);

menuCanvas.SetActive(false);

gameplayCanvas.SetActive(false);

changeLevelCanvas.SetActive(false);

levelStageCanvas.SetActive(false);

completeGameCanvas.SetActive(false);

}

}

* 1. **LevelManager.cs:**

using UnityEngine;

using UnityEngine.UI;

public class LevelManager : MonoBehaviour

{

public const int numLevel = 30; // Tổng số lượng Level trong game (hằng số)

public int currentLevel = 1; // Level hiện tại

public GameObject levelText;

void Update()

{

levelText.GetComponent<Text>().text = "Level " + currentLevel;

}

public int CurrentLevel

{

get { return currentLevel; }

set

{

if (value >= 1 && value <= numLevel) // Giới hạn giá trị trong khoảng hợp lệ

{

currentLevel = value;

}

else

{

Debug.LogWarning("Giá trị CurrentLevel không hợp lệ!");

}

}

}

public int NumLevel

{

get { return numLevel; }

}

// Level tốt nhất đạt được (lưu trữ bằng PlayerPrefs)

public int BestLevel

{

get { return PlayerPrefs.GetInt("BestLevel", 1); }

set

{

if (value >= 1 && value <= numLevel && value > BestLevel)

{

PlayerPrefs.SetInt("BestLevel", value);

PlayerPrefs.Save();

}

}

}

// Singleton để đảm bảo chỉ có một instance của LevelManager

public static LevelManager Instance;

private void Awake()

{

if (Instance == null)

{

Instance = this;

DontDestroyOnLoad(gameObject); // Không phá hủy LevelManager khi chuyển Scene

}

else

{

Destroy(gameObject); // Đảm bảo chỉ có một instance

}

}

// Hàm kiểm tra và cập nhật BestLevel

public void UpdateBestLevel()

{

if (currentLevel > BestLevel)

{

BestLevel = currentLevel;

}

}

// Reset BestLevel (nếu cần)

public void ResetBestLevel()

{

PlayerPrefs.DeleteKey("BestLevel");

PlayerPrefs.Save();

}

}

* 1. **QuestionDisplay.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class QuestionDisplay : MonoBehaviour

{

public GameObject screenQuestion;

public static string newQuestion;

// Update is called once per frame

void Update()

{

StartCoroutine(PushTextOnScreen());

}

IEnumerator PushTextOnScreen()

{

yield return new WaitForSeconds(0.01f);

screenQuestion.GetComponent<Text>().text = newQuestion;

}

}

* 1. **QuestionGenerate.cs:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class QuestionGenerate : MonoBehaviour

{

public static string actualAnswer; // Đáp án đúng

public static string currentQuestion; // Câu hỏi hiện tại

public static string currentQuestionID; // ID của câu hỏi hiện tại

public static bool displayQuestion = false;

private List<string> questions = new List<string>(); // Danh sách câu hỏi

private List<string> answers = new List<string>(); // Danh sách đáp án

private List<string> questionIDs = new List<string>(); // Danh sách ID câu hỏi

void Start()

{

LoadQuestionsFromFile(); // Đọc dữ liệu từ file

//GenerateQuestion(); // Tạo câu hỏi cho currentLevel

}

void Update()

{

if (CanvasManager.Instance.gameplayCanvas.activeSelf)

{

if (!displayQuestion)

{

GenerateQuestion(); // Chỉ tạo câu hỏi khi chưa hiển thị

}

}

}

void LoadQuestionsFromFile()

{

// Tải file từ Resources

TextAsset questionData = Resources.Load<TextAsset>("questions"); // Không cần .txt

if (questionData != null)

{

string[] blocks = questionData.text.Split(new string[] { "===" }, System.StringSplitOptions.RemoveEmptyEntries); // Chia theo `===`

foreach (string block in blocks)

{

string[] lines = block.Trim().Split('\n'); // Lấy từng dòng trong một block

string questionID = ""; // Chỉ số câu hỏi

string question = ""; // Nội dung câu hỏi

string answer = "";

for (int i = 0; i < lines.Length; i++)

{

if (i == 0) // Dòng đầu tiên là ID câu hỏi

{

questionID = lines[i].Trim();

}

else if (lines[i].Contains("|")) // Dòng chứa đáp án

{

answer = lines[i].Split('|')[1].Trim(); // Lấy phần sau dấu `|`

}

else

{

question += lines[i] + "\n"; // Thêm vào nội dung câu hỏi

}

}

if (!string.IsNullOrEmpty(questionID) && !string.IsNullOrEmpty(question) && !string.IsNullOrEmpty(answer))

{

questionIDs.Add(questionID); // Lưu ID câu hỏi

questions.Add(question.Trim());

answers.Add(answer);

}

}

}

else

{

Debug.LogError("Không tìm thấy file questions.txt trong Resources.");

}

}

void GenerateQuestion()

{

int currentLevel = LevelManager.Instance.CurrentLevel; // Lấy Level hiện tại từ LevelManager

if (currentLevel > 0 && currentLevel <= questions.Count)

{

int index = currentLevel - 1; // Chỉ số trong danh sách bắt đầu từ 0

Debug.Log(currentLevel);

// Lấy câu hỏi, ID, và đáp án theo currentLevel

currentQuestionID = questionIDs[index];

currentQuestion = questions[index];

actualAnswer = answers[index];

// Hiển thị câu hỏi

QuestionDisplay.newQuestion = $"{currentQuestion}";

displayQuestion = true;

}

else

{

Debug.LogError($"Level {currentLevel} vượt quá số lượng câu hỏi ({questions.Count}).");

}

}

public void OnCorrectAnswer()

{

displayQuestion = false;

// Cập nhật bestLevel nếu vượt qua level hiện tại

LevelManager.Instance.UpdateBestLevel();

}

}

**CHƯƠNG III. CÁC CÔNG NGHỆ VÀ TÀI NGUYÊN SỬ DỤNG**

1. **Xây dựng và triển khai game bằng Unity:**

Trong quá trình xây dựng trò chơi, Unity được chọn làm nền tảng phát triển chính nhờ vào tính linh hoạt, dễ sử dụng và khả năng hỗ trợ đa nền tảng. Unity cung cấp một môi trường tích hợp mạnh mẽ với hệ thống kéo-thả trực quan, giúp giảm thiểu độ phức tạp khi thiết kế giao diện và logic game. Các công cụ mạnh mẽ như **Canvas** để quản lý giao diện người dùng, **Physics Engine** để xử lý tương tác vật lý, và **Animation System** để tạo ra các hiệu ứng chuyển động mượt mà đã được tận dụng tối đa trong quá trình phát triển. Với Unity, việc thử nghiệm và triển khai trò chơi trên nhiều nền tảng như Android và Windows trở nên dễ dàng, đảm bảo tính ổn định và trải nghiệm nhất quán cho người chơi.

A screenshot of a computer

Description automatically generated

“Giao diện Unity”

1. **Vẽ biểu đồ UML bằng StarUML:**

Trong quá trình phát triển trò chơi, StarUML được sử dụng như một công cụ hỗ trợ quan trọng để xây dựng các biểu đồ UML nhằm mô tả cấu trúc và hành vi của hệ thống. Với giao diện trực quan và dễ sử dụng, StarUML giúp tạo ra các biểu đồ như Use Case, Class Diagram, và Sequence Diagram, hỗ trợ phân tích và thiết kế chi tiết các thành phần trong game. Công cụ này cho phép biểu diễn mối quan hệ giữa các lớp, chức năng của người chơi, và luồng xử lý của hệ thống một cách rõ ràng và logic. Nhờ khả năng tùy chỉnh linh hoạt và hỗ trợ nhiều loại biểu đồ khác nhau, StarUML giúp đảm bảo quá trình phát triển game được tổ chức một cách khoa học và dễ dàng quản lý.

A screenshot of a computer

Description automatically generated

“Giao diện StarUML”

1. **Sử dụng các asset có sẵn trên itch.io:**

Trong quá trình phát triển trò chơi, các tài nguyên có sẵn từ itch.io được tận dụng để tiết kiệm thời gian và nâng cao chất lượng sản phẩm. Itch.io là một nền tảng phong phú, cung cấp nhiều asset miễn phí và trả phí bao gồm hình ảnh, âm thanh, nhạc nền, và các mẫu thiết kế giao diện. Các asset này không chỉ giúp trò chơi trở nên sinh động hơn mà còn giảm bớt công sức cho việc tạo nội dung từ đầu. Việc sử dụng tài nguyên từ itch.io cũng đảm bảo tính chuyên nghiệp và đa dạng cho trò chơi, đồng thời tạo điều kiện để tập trung vào việc phát triển các tính năng chính và tối ưu hóa trải nghiệm người chơi.

**CHƯƠNG IV: THIẾT KẾ VÀ TRIỂN KHAI SẢN PHẨM**

1. **Thiết kế sản phẩm:**
   1. **Giao diện Intro:**

**A close-up of a white background

Description automatically generated**

* 1. **Giao diện Menu:**

**A screenshot of a game

Description automatically generated**

* 1. **Giao diện Gameplay:**

**A screenshot of a black screen

Description automatically generated**

* 1. **Giao diện Change Level:**

**A screenshot of a computer

Description automatically generated**

* 1. **Giao diện Complete Game:**

**A screenshot of a computer

Description automatically generated**

1. **Triển khai sản phẩm:**

Game được triển khai dưới dang file apk cho Android. Việc triển khai trò chơi trên nền tảng Android được thực hiện nhằm đảm bảo tính phổ biến và khả năng tiếp cận cao đối với người dùng. Quy trình triển khai bao gồm các bước quan trọng như xuất bản trò chơi từ Unity dưới dạng tệp APK và thực hiện kiểm tra trên các thiết bị Android.

Đầu tiên, trò chơi được tối ưu hóa để đảm bảo khả năng chạy mượt mà trên nhiều loại thiết bị Android với cấu hình khác nhau. Điều này bao gồm việc tinh chỉnh chất lượng đồ họa, quản lý tài nguyên, và kiểm soát kích thước tệp trò chơi.

Tiếp theo, Unity cung cấp công cụ tích hợp để xây dựng trò chơi dưới định dạng **Android Build**. Sau khi hoàn tất xây dựng, tệp APK được tạo ra và cài đặt thử nghiệm trên các thiết bị thực tế để kiểm tra tính ổn định, giao diện người dùng, và khả năng tương tác.

Sản phẩm có thể chạy ổn định trên các thiết bị android khác nhau hoặc giả lập android trên laptop.

**CHƯƠNG V: TỔNG KẾT**

Trò chơi đã hoàn thành đúng theo kế hoạch đề ra, mang đến một trải nghiệm giải đố thú vị và bổ ích cho người chơi. Trong suốt quá trình xây dựng, từ khâu ý tưởng, thiết kế, đến triển khai, em đã không ngừng học hỏi và cải tiến để tạo ra sản phẩm cuối cùng vừa đáp ứng được yêu cầu kỹ thuật, vừa đảm bảo tính sáng tạo và hấp dẫn.

Trong tương lai, trò chơi sẽ được phát triển thêm nhiều nội dung mới, bao gồm các câu đố phong phú hơn với nhiều cấp độ khó khác nhau, hệ thống bảng xếp hạng trực tuyến để người chơi so tài, và tích hợp thêm tính năng chia sẻ kết quả qua mạng xã hội. Bên cạnh đó, đồ họa và giao diện sẽ tiếp tục được nâng cấp để tối ưu trải nghiệm người dùng, đồng thời trò chơi sẽ được phát hành trên nhiều nền tảng khác như iOS, Windows, và web để tiếp cận thêm nhiều đối tượng người chơi.

Quá trình thực hiện dự án đã mang lại nhiều bài học quý giá. Em học được cách tổ chức và quản lý dự án phần mềm một cách khoa học, từ việc xây dựng cấu trúc mã nguồn đến tối ưu hóa hiệu suất. Những thách thức gặp phải trong quá trình phát triển, như khắc phục lỗi kỹ thuật hay tìm ra giải pháp cho các tính năng mới, đã rèn luyện khả năng giải quyết vấn đề một cách sáng tạo và hiệu quả. Ngoài ra, việc sử dụng các công cụ như Unity, StarUML, và khai thác tài nguyên từ itch.io cũng giúp em nâng cao kỹ năng làm việc với các công cụ và tài nguyên hiện có.

Kết quả đạt được không chỉ là một trò chơi hoàn thiện, mà còn là nền tảng vững chắc cho những dự án tương lai, cùng với sự tự tin và kiến thức tích lũy qua từng bước phát triển. Chúng tôi hy vọng trò chơi sẽ mang lại niềm vui cho người chơi và là động lực để tiếp tục phát triển các sản phẩm sáng tạo hơn trong tương lai.

**CHƯƠNG VI: TÀI LIỆU THAM KHẢO VÀ LIÊN KẾT**

Link project: <https://github.com/quangthien1310/quizgame>

Báo cáo word: <https://github.com/quangthien1310/quizgame/blob/master/Project3.docx>

Biểu đồ UML: <https://github.com/quangthien1310/quizgame/blob/master/project3.mdj>

Link asset: <https://prinbles.itch.io/yet-another-icons>