Running to reality

1. Project Overview

This is a simple 2D endless runner where the player jumps over obstacles to survive. The goal is to last as long as possible and increase the score by successfully jumping over obstacles. As the player progresses, the speed increases, making the game more challenging.

2. Project Review

This project is inspired by the **Dino Run** game, which appears when the internet connection is lost. To enhance engagement, the game introduces **three selectable themes**, each reflecting different aspects of life's journey.

The overarching concept is based on **"หลักสัจธรรมของชีวิต"** (The Fundamental Truths of Life):

- **Birth** (**Escaping F**) – After being born and growing up, we enter the student phase. The main challenge of this stage is **escaping an F grade**, symbolizing the struggles of academic life.

- **Aging & Illness** (**Escaping T**) – Time is limited, and as we grow older, responsibilities increase. This theme represents the **struggle against time**, where every moment counts.

- **Death** (**Escaping G**) – The inevitable end of life is symbolized by the ghost theme, representing the **inability to escape fate**.

These themes transform the traditional endless runner into a meaningful experience, blending fun with life reflections.

Programming Development

3.1 Game Concept

The game is an endless runner where players control a character that automatically moves forward, jumping to avoid obstacles. The goal is to survive as long as possible while earning points based on skillful jumps.

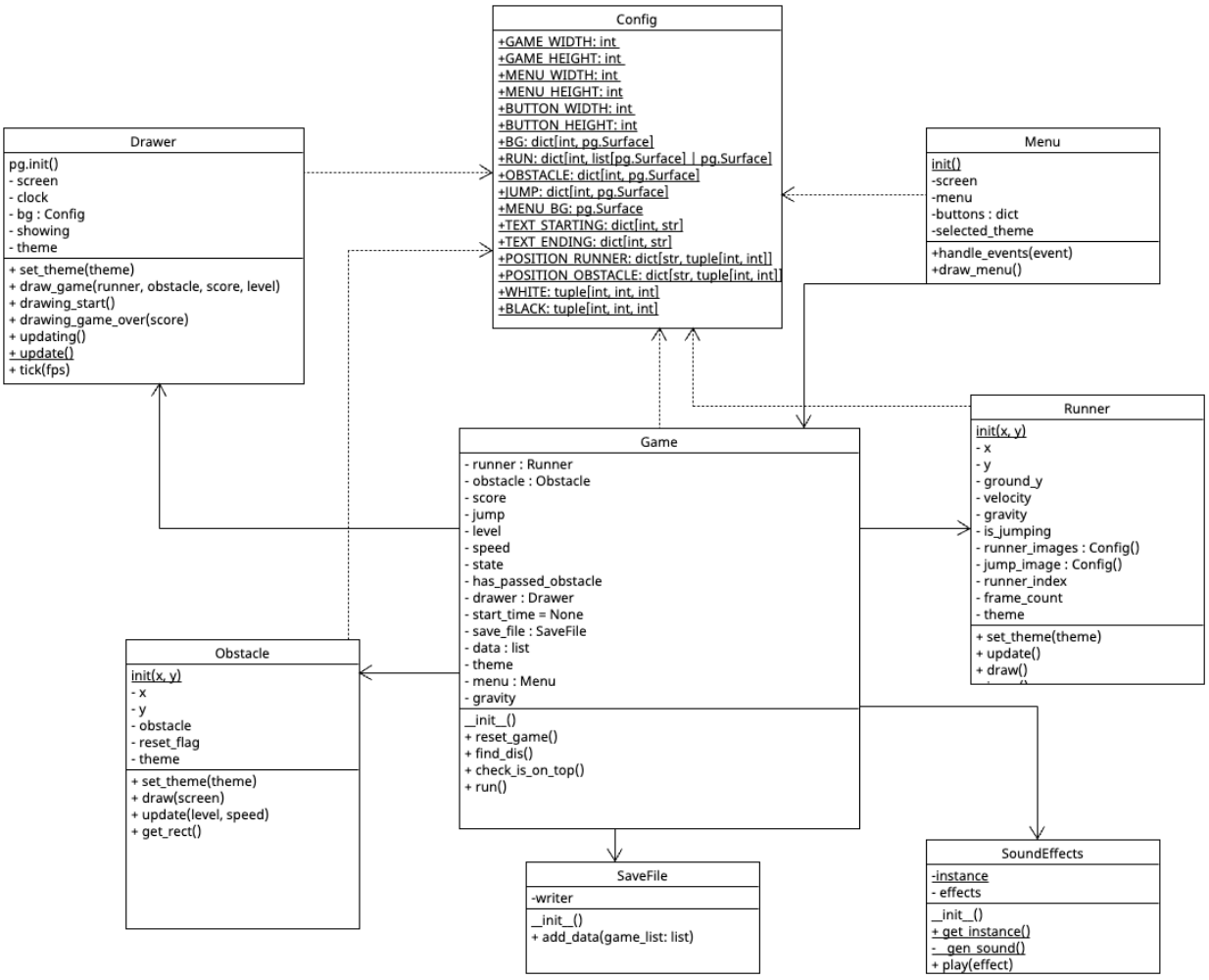
Key Features:

- Jump Mechanic: Press spacebar to jump.
- Obstacle Spawning: Obstacles move from right to left.
- Collision Detection: Hitting an obstacle ends the game.
- Score System: Tracks jumping over obstacles.
- Game Over and Final Score Screen: Displays when the player loses.
- Score Screen: Display when during the game.

3.2 Object-Oriented Programming Implementation

The game consists of **five main classes**:

1. **Game** – The core class that controls the entire game loop, including updates, rendering, and managing interactions between other classes.
2. **Drawer** – Handles visual elements like displaying the score, "Game Over" messages.
3. **Obstacle** – Responsible for generating and managing obstacles that the player must avoid.
4. **Runner** – Responsible for generating runner animation
5. **Sound** – Manages game audio, including background music and sound effects.
6. **Config** – Stores all configuration details, such as file paths for images, sounds, and other game settings.
7. **`Save file`** – Records and stores gameplay statistics for analysis.
8. **Menu** – Handle the menu page.



3.3 Algorithms Involved

- **Collision Detection** – Uses bounding box collision detection to check if the player collides with obstacles.
- **Score Progression** – Increases the difficulty (speed) as the score rises.

3. Statistical Data (Prop Stats)

4.1 Data Features

Game Features:

1. **Score System** – The player's score increases when they successfully jump past an obstacle.
2. **Level Progression** – When the score reaches **10**, the level increases, and the player's speed also increases.
3. **Theme Selection** – Players can choose a theme before the game starts, customizing the background and visuals.
4. **Time Tracking** – The game records the total time from when the player starts until the game ends.
5. **Jump Counter** – Tracks the total number of jumps made by the player throughout the game.
6. **Speed Tracking** – Logs the player's final game speed and speed progression over time.

4.2 Data Recording Method

The game records data using a CSV file. This method is used to store gameplay statistics that are:

- **Score** – The player's final score.
- **Level** – The highest level reached.
- **Theme** – The selected theme before the game starts.
- **Time Played** – The total duration of the gameplay session.
- **Total Jumps** – The number of jumps made by the player.
- **Speed** - The latest speed before the game is over.

4.3 Data Analysis Report

Tables – Display the 5 rows of the data.
Bar Charts – Show level progression and score distribution.
Line Graphs – Track player performance over time.

Scatter Plots – Analyze the relationship between jumps and score.

	Why is it good to have this data? What can it be used for	How will you obtain 50 values of this feature data?	Which variable and which class will you collect this form	How will you display this feature data
score	Tracks progress and performance	Play until game over	self.__score in Game class	histogram
level	Shows how far players get before failing	Play until game over	self.__level in Game class	histogram
theme	Analyzes player preferences	Play until game over	self.__theme in Game class	Pie graph
time played	Measures engagement.	Play until game over	self.__start_time - time.time in Game class	histogram
total jump	Evaluates player skill	Play until game over	self.__jump in Game class	histogram
speed	Helps balance difficulty	Play until game over	self.__speed in Game class	histogram/ Line graph

	Feature Name	Graph objective	Graph type	x-axis	y-axis
graph1	Total jump	Show the distribution	Histogram	Total jump	Frequency
graph2	Theme	Show the proportion of selected theme	Pie graph	The total of selected theme in percentage	
graph3	Score, Theme	Show the distribution of the score via each theme	Box plot	Score	Theme
graph4	Total jump, Time played	Show to correlation between total jump and time played	Scatter plot	Total Jump	Time played

4. Project Timeline

Week	**Task**
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Submission the data analytic part

4 (31 March)	Submission the data analytic part
5 (7 April)	
6 (14 April)	Submission week (Draft)

5. Document version

Version: *1.0*
Date: 31 March 2025

Date	Name	Description of Revision, Feedback, Comments
14/3	Rattapoom	Good Job!
16/3	Parima	The proposal is clear and great in detail.
29/3	Rattapoom	You'd have to play the game 50 times to collect data. This may not be a problem since the original Dinosaur Running game doesn't take long to complete (unless you're very good at it). You might want to consider changing the attributes that you're going to collect or writing an autoplay bot to help you collect data.
29/3	Parima	The UML diagram may need some revision, but the overall is great. Good Work!