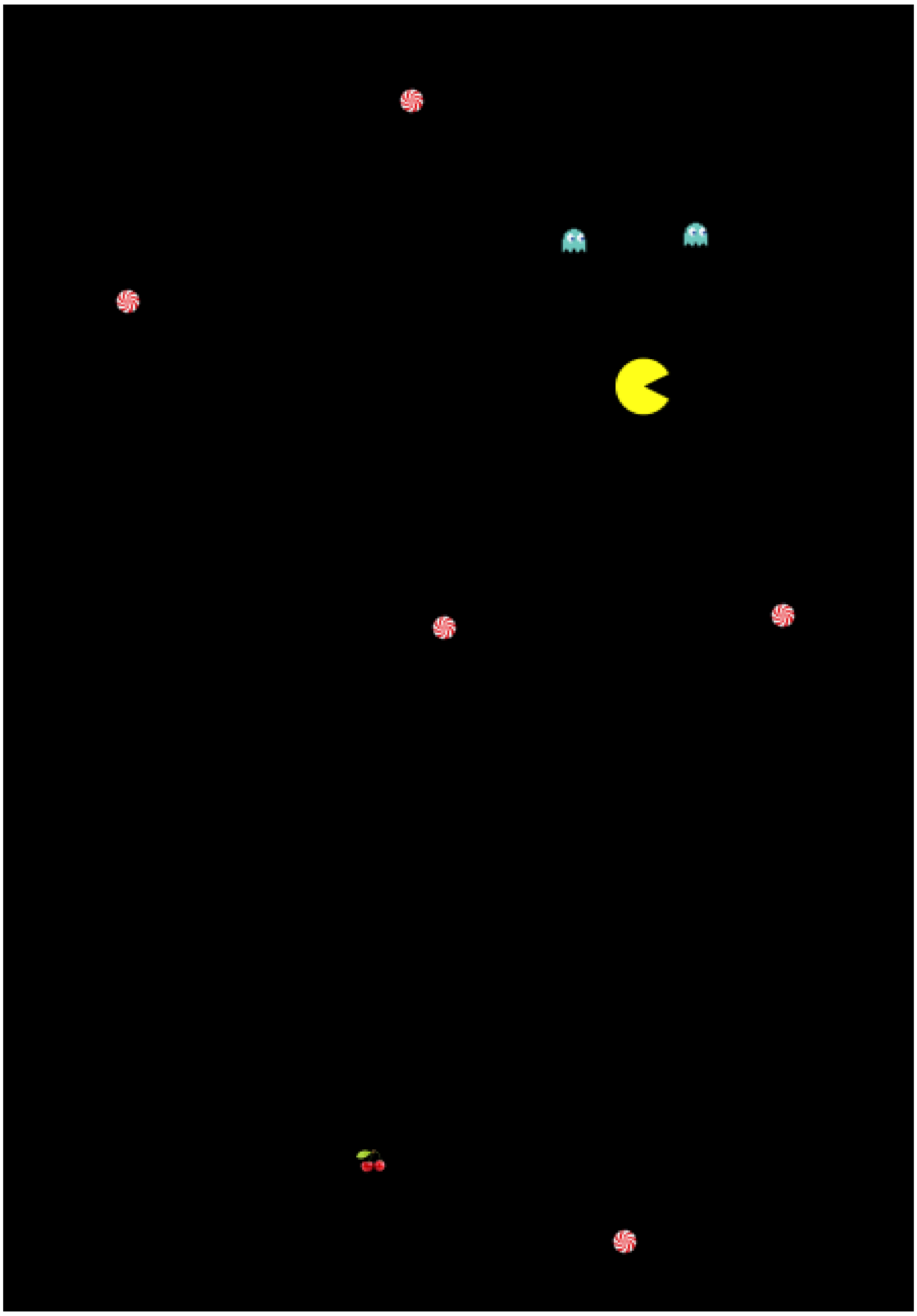


DASHING CANDIES!

Seed sowed by: Melissa VanderLely

Watered and groomed by: Dhaval Tailor

Game Overview:

DASHING CANDIES is an arcade-style 2D grid-based game where you control a character that moves around the grid to collect dots while avoiding enemies. The more dots you collect, the bigger your character grows, making it harder to navigate the grid and avoid collisions. However, fruits appear intermittently to reset your size, offering you a chance to shrink back down and improve your manoeuvrability.

The game has multiple levels of increasing difficulty, and each level adds new challenges. Your goal is to collect as many dots as possible to increase your score, all while dodging enemies and managing your size.

Purpose: The application is designed to improve hand-eye coordination and strategic movement by challenging the player to balance between avoiding enemies and collecting candies. Subtly also encouraging players to consume fruits to keep healthy.

Key Features:

- Player movement and size management
- Random enemy and fruit spawn
- Collision detection
- Multiple levels with increasing difficulty

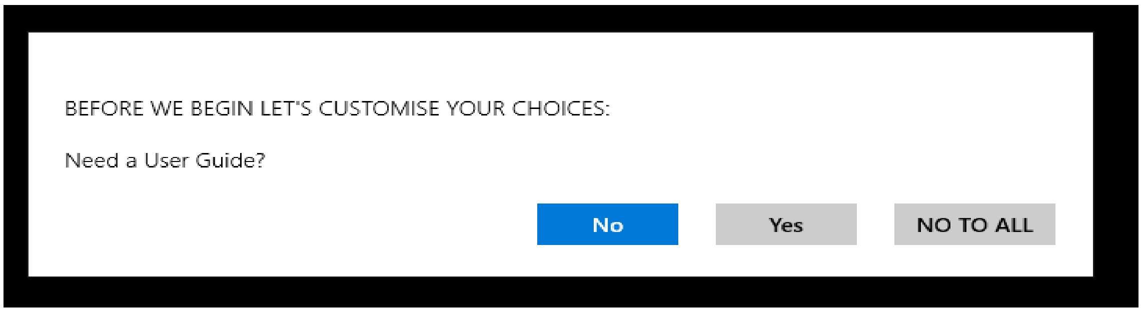
Target Audience: This game targets users interested in casual but challenging games, particularly those looking for quick and engaging gameplay sessions.

User Guide: Access this guide for instructions

Welcome to **DASHING CANDIES!** This guide will walk you through the gameplay, objectives, controls, and strategies to help you get the most out of your game experience. Be sure to read through the entire guide so you're fully prepared before jumping into the action!

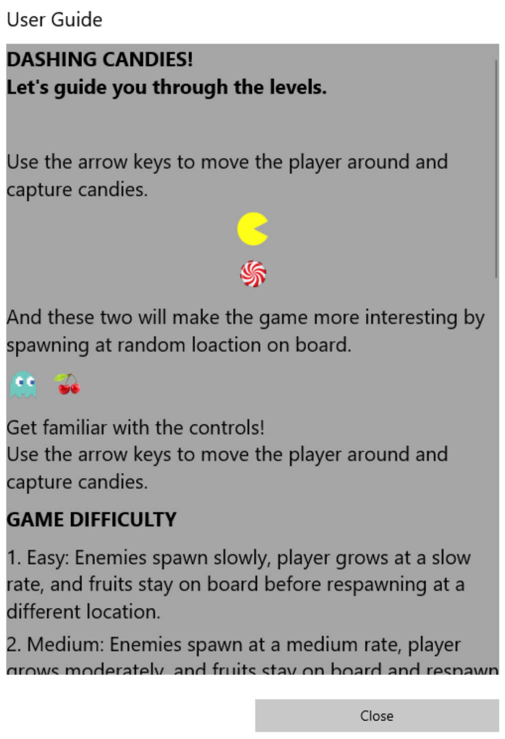
Menu Options:

Before starting the game, you'll be presented with a main menu where you can choose from several options:


BEFORE WE BEGIN LET'S CUSTOMISE YOUR CHOICES:

Need a User Guide?

- **No:** Choose "No" if you don't need the User guide before the game.
- **Yes:** Choose "Yes" if you need the User guide before the game.


User Guide

DASHING CANDIES!
Let's guide you through the levels.

Use the arrow keys to move the player around and capture candies.

And these two will make the game more interesting by spawning at random loaction on board.

Get familiar with the controls!
Use the arrow keys to move the player around and capture candies.

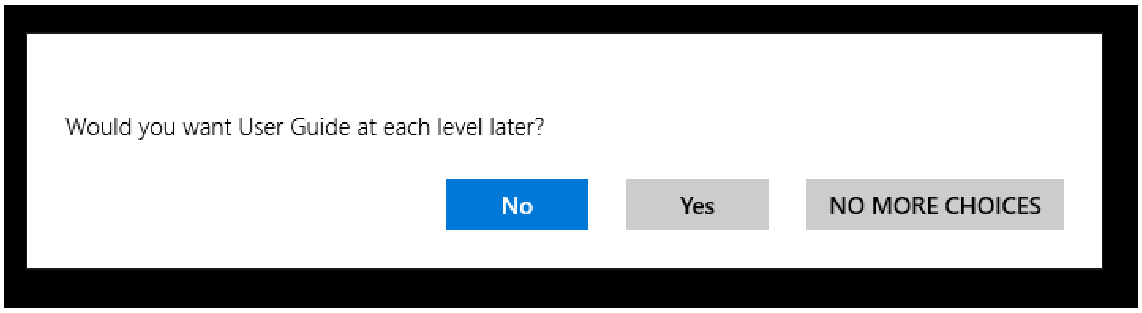
GAME DIFFICULTY

1. Easy: Enemies spawn slowly, player grows at a slow rate, and fruits stay on board before respawning at a different location.

2. Medium: Enemies spawn at a medium rate, player grows moderately, and fruits stay on board and respawn

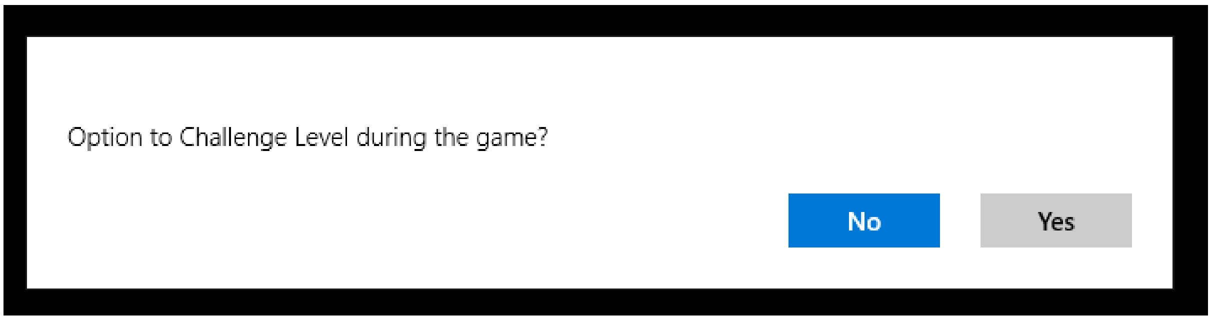
- **NO TO ALL:** Choose "NO TO ALL" if you don't want to customise choices.

After choosing "No" or closing the guide, screen will be prompted for


Would you want User Guide at each level later?

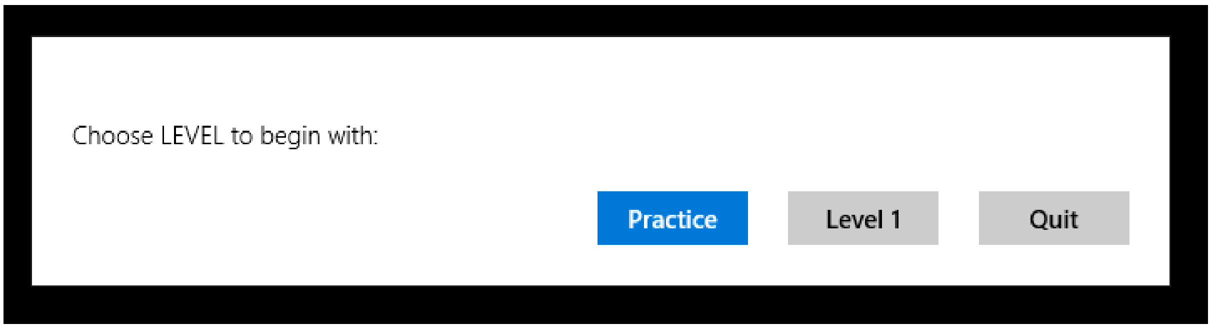
- **No:** Choose “No” if you don't want to see a brief User guide before every level.
- **Yes:** Choose “Yes” if you need the brief User guide will popup before every level.
- **NO MORE CHOICES:** Choose “NO MORE CHOICES” if you don't want to customise choices anymore.

After choosing “No” or ”Yes”, screen will be prompted for


Option to Challenge Level during the game?

- **No:** Choose “No” if you don't intend to change the level of challenge at each level.
- **Yes:** Choose “Yes” if you intend to change the level of challenge at each level.

At any point if you choose “NO TO ALL” OR “NO MORE CHOICES” or either of the choices in the previous prompt, screen will be prompted for

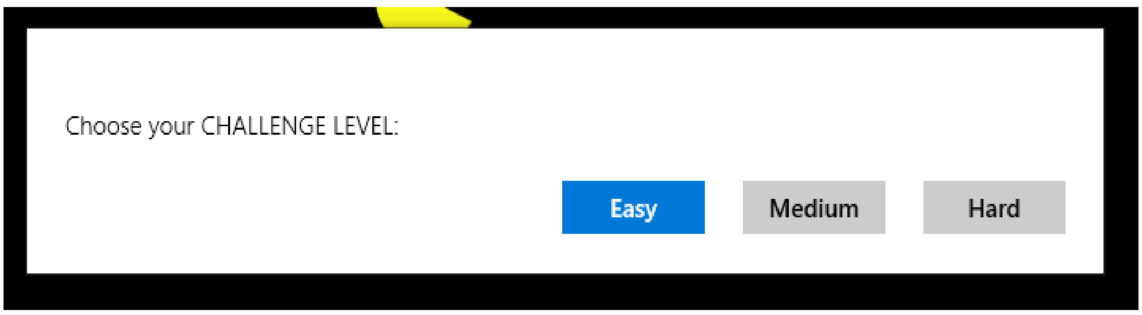

Choose LEVEL to begin with:

Here is where the game begins.

We will discuss “Practice” and “Level 1” later.

- **Quit**: Will Exit the game at any level

Depending on your choices you made, you would have a challenge level to pick from the below mentioned three choices.



ChallengeLevels:

The game features three difficulty levels, each offering a progressively tougher challenge. Choose wisely based on your skill level!

1. **Easy**:
   - Enemies spawn slowly, allowing you more time to collect dots.
   - Your character grows at a slow rate, making it easier to navigate the grid.
   - Fruits appear often and remain on the grid longer, giving you more chances to reset your size.
2. **Medium**:
   - Enemies spawn more frequently, forcing you to plan your moves carefully.
   - Your character grows at a moderate pace as you collect dots.
   - Fruits appear less frequently and disappear faster, so you must act quickly to reset your size.
3. **Hard**:
   - Enemies spawn rapidly, making it difficult to avoid collisions.
   - Your character grows quickly, increasing the challenge of moving around the grid.
   - Fruits appear very rarely and disappear quickly, leaving you little time to reduce your size.
   - A **game timer** limits how long you can play, adding extra pressure to collect dots quickly.

Game Progression:

The game is divided into different levels, each introducing new challenges:

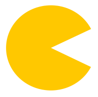
- **Practice:**
    - Just to get familiar with the controls.
    - Score is not considered during practice.
- **Level 1**:
    - Slow-paced with fewer enemies.
    - Candies and fruits are easier to collect, and your character grows slowly.
- **Level 2**:
    - Moderate difficulty with faster enemy spawn rates.
    - Candies are still easy to collect, but fruits are less frequent, so you must balance size management and enemy avoidance.
- **Level 3**:
    - High difficulty with rapid enemy spawns and fast player growth.
    - Fruits disappear quickly, and the game has a time limit, adding to the pressure.

The player advances through levels by achieving specific scores or surviving for a certain amount of time.

Game Controls:

The controls in **DASHING CANDIES** are simple but essential for mastering the game:

- **Arrow Keys**:
    - **Up Arrow**: Move the player character upward.
    - **Down Arrow**: Move the player character downward.
    - **Left Arrow**: Move the player character to the left.
    - **Right Arrow**: Move the player character to the right.

 = YOU IN THE GAME

These controls allow you to move around the game grid and collect candies (dots in earlier versions) or avoid enemies. Mastering movement is critical to surviving in the game, especially in harder difficulty levels where enemies spawn more frequently.

Gameplay Mechanics:

1. Collecting Candies: (dots in earlier version)

- **Candies** are the primary objects you'll be collecting to increase your score. Each dot consumed adds 1 point to your score, but also increases the size of your character. As your character grows larger, it becomes more difficult to navigate the grid and avoid enemies. Think carefully about which dots to collect and when to move toward fruits to reset your size.

2. Fruits:

- **Fruits** appear randomly on the grid and act as a size reset for your character. If you collect a fruit, your character's size will revert to its original small size, making it easier to move around and avoid enemies. However, fruits only appear for a limited time, and their frequency decreases with higher difficulty levels. You must act quickly when a fruit appears!

3. Enemies:

- **Enemies** are the obstacles that make the game challenging. They spawn randomly on the grid and move toward or away from the player (depending on the game mode). Avoiding enemies becomes harder as your character grows in size, so resetting your size with fruits is crucial to survival. If your character collides with an enemy, the game ends immediately.

Collision Detection:

- The game detects when your character collides with candies, fruits, enemies, or borders.
  - **Candies**: Increase your score and your size (depending on level).
  - **Fruits**: Reset your size.
  - **Enemies**: End the game if collided with.
  - **Borders**: Prevent movement beyond the game grid.

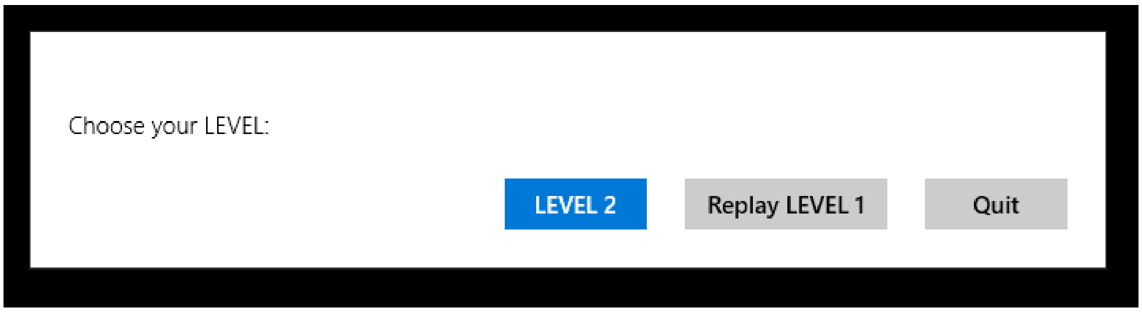
Scoring:

The game's scoring system is straightforward:

- Each **candy** you collect increases your score by 1 point.
- Your final score is displayed at the end of the game. If you reach a new high score, it will be saved and displayed in future game sessions.

Your high score is saved across sessions, so you can always try to beat your personal best!

End Game or Replay:



When the game ends (either by colliding with an enemy or running out of time in timed levels), a dialog will appear displaying your final score. You will have the option to:

- **Replay**: Restart the game and try to beat your previous score.
- **Quit**: Exit the game.

Tips for Playing:

Here are some tips to help you succeed in **DASHING CANDIES**:

1. **Focus on Fruits**: The key to staying small and nimble is to prioritize collecting fruits. Always keep an eye out for fruits appearing on the grid, especially as your character grows larger.
2. **Manage Your Size**: While it's tempting to collect every candy, remember that growing too large will make it harder to avoid enemies. Plan your movements strategically and aim for fruits to reset your size when needed.
3. **Avoid Enemies**: Enemies are your biggest threat. Always plan your movements in advance to avoid getting trapped between enemies, especially when your size increases.

4. **Use Borders Wisely**: The borders prevent you from moving off the screen but can also be used to your advantage. Trapping enemies near the edges of the screen can create space for you to collect dots more safely.
5. **Start on Easy**: If you’re new to the game, start on **Easy** difficulty. It gives you more time to get used to the controls and mechanics before increasing the challenge.
6. **Watch the Timer**: In harder levels, the game timer adds extra pressure. Focus on collecting dots efficiently and avoid unnecessary risks with enemies.

With this **User Guide**, you now have a full understanding of the game mechanics, objectives, and controls. Remember to use strategy as you navigate through the levels, manage your size, and avoid enemies. Good luck, and have fun playing **DASHING CANDIES**!

TECHNICAL DOCUMENTATION:

Let's take a deep dive into the technical aspects.

Technical Specifications:

- Developed using UWP and C#
- Implements IComparable<GamePiece> and IEquatable<GamePiece>
- Uses timers for game mechanics (e.g., fruit spawn, game duration)

Scope:

- Implementation of player, enemy, candy, and fruit mechanics
    - Player movement and collision detection
    - Dynamic spawning of game elements (dots, fruits, enemies)
    - Score tracking and display
    - Game mode selection
- Multiple difficulty levels with different spawn rates and player behaviours
    - This reflects with the actual development of the game

Class Descriptions and Diagrams for all classes

Namespace: GameLibrary

- This namespace contains the classes and interfaces that define the game logic, specifically the GamePiece class, which represents a movable game object with graphical representation.

GamePiece Class: Represents game objects (player, dots, enemies, and fruits) and provides methods for movement, collision detection, and location updates.

Key Fields:

- **private Thickness objectMargins**:
  This field stores the position of the game piece on the board, using margins to define the location relative to the grid. Margins are used instead of X-Y coordinates in UWP apps.
- **private Image onScreen**:
  This field stores the image representation of the game piece. The image is displayed on the game grid.
- **private RotateTransform rotate**:
  Handles the rotation of the game piece’s image. The rotation is required when the game piece changes direction (e.g., the player moves up, down, left, or right).

- **private static Random random**:
  A shared instance of Random used to generate random positions or other random properties (like candies, enemy and fruits spawn locations).

Key Properties:

- **public Thickness Location { get; }**:
  Read-only property that returns the current location (margins) of the game piece.
- **public Image Image { get; }**:
  Read-only property that returns the Image object associated with the game piece, used to display it on the grid.
- **public int Points { get; set; }**:
  Represents the score or value of the game piece. For example, dots might have a point value that increases the player's score when consumed.
- **public int Size { get; set; }**:
  Represents the size of the game piece. This is used for scaling purposes, such as enlarging the player when dots are consumed.

Key Methods:

- **public GamePiece(Image img, int point = 1)**:
  Constructor that initialises the game piece with an image and optional point value. It sets the image margins and rotation centre for future movements.
- **private void OnImageSizeChanged(object sender, SizeChangedEventArgs e)**:
  Event handler that adjusts the rotation centre of the game piece when the image size changes. This ensures that the image remains correctly rotated around its centre.
- **public bool Move(Windows.System.VirtualKey direction)**:
  Moves the game piece in a specified direction (up, down, left, right) based on the virtual key pressed by the user. It also rotates the game piece to face the direction of movement.
- **public int CompareTo(GamePiece other)**:
  Implements IComparable<GamePiece>. Compare two game pieces based on their vertical (Top margin) position first, and then their horizontal (Left margin) position if they are on the same vertical plane. This is used for sorting or collision detection.
- **public bool Equals(GamePiece other)**:
  Implements IEquatable<GamePiece>. Checks if two game pieces are at the same position on the board by comparing their margins.
- **public void NewLocation(Thickness newLocation)**:
  Updates the game piece's position to a new location by setting the margins accordingly.

Namespace: GameInterface

The GameInterface namespace is responsible for managing the user interface and overall game flow. It contains classes that handle player input, manage the game state, display game elements on the screen, and control interactions between game objects such as the player, dots, fruits, and enemies.

Key classes in this namespace include:

- **MainPage**: The main controller class that initialises the game, handles user input (e.g., player movement through keyboard controls), and manages interactions between game elements such as collisions and scoring. It also contains methods for starting, resetting, and timing the game.
- **NotMain**: A class that stores game settings and logic for managing difficulty levels, timers, and high scores. It controls game states such as when the game ends or progresses to new levels and handles the spawning of enemies, dots, and fruits.

Together, these classes provide the framework for the game's user experience, ensuring smooth transitions between gameplay, menus, and various game events.

MainPage Class: Handles the game's UI and logic, including player movement, game timer, and collision detection.

Key **Fields:**

- **private NotMain nM:**
  Holds an instance of the NotMain class, which contains game settings and other configurations.
- **public readonly Random placement:**
  A random object used for determining the random placement of game pieces (dots, fruits, enemies) on the grid.
- **public ScrollViewer practiceScrollViewer:**
  A ScrollViewer used when displaying the user guide or help sections.

Key Methods:

- **public MainPage():**
  Constructor that initialises the MainPage and sets up various components, such as event listeners for key inputs and dimensions of the game screen.
- **public GamePiece PieceUniqueSpot(string imgSrc, double size, List<GamePiece> g):**
  Generates a game piece at a unique random position on the board, ensuring it does not overlap with any existing pieces. This method is used to spawn dots, enemies, and fruits.

- **public void ResetGame():**
  Resets the game state by clearing the grid of all game pieces, resetting the score, and stopping any timers.
- **public void GameTimerEffet(object sender, object e):**
  Manages the game's timer and handles actions when the timer reaches certain points (e.g., end of game, removing game pieces).
- **public GamePiece CreatePiece(string imgSrc, double size, int left, int top):**
  Creates a new game piece with the specified image and size, placing it at the provided left and top position on the grid.
- **public async void FruitTimerEffect(object sender, object e):**
  Handles the fruit respawn timer. When a fruit disappears, a new fruit is spawned at a random location.
- **public void CollideDots():**
  Checks for collisions between the player and dots. If a collision is detected, the dot is consumed, and the player's size increases.
- **public void CollideFruits():**
  Checks for collisions between the player and fruits. If a fruit is consumed, the player's size resets to the original size.
- **public void CollideEnemy():**
  Checks for collisions between the player and enemies. If a collision occurs, the game ends, and the player loses.
- **public void CoreWindow_KeyDown(CoreWindow sender, KeyEventArgs e):**
  Handles keypresses for controlling player movement. It also checks for collisions with borders, dots, fruits, and enemies.
- **public void PlayGame(bool enableEnemy = false, bool enableFruits = false, bool enableGameTimer = false):**
  Starts the game with the specified settings (e.g., whether to enable enemies, fruits, or a game timer).

NotMain Class: Manages game settings such as difficulty, game modes, and interaction with timers for fruits and enemies.

Key Fields:

- **public int dotCount**:
  Defines how many dots are placed on the board for collection by the player.
- **public double gT**:
  Represents the game’s time limit (in minutes) for certain levels.
- **public bool gameOn**:
  A boolean flag that indicates whether the game is currently running.

- **public List<GamePiece> manyDots, manyEnemies, manyFruits, borders**:
  Collections that hold the various game pieces (dots, enemies, fruits, and borders) present on the board.

Key Methods:

- **public void Easy()**:
  Sets the game difficulty to "Easy," adjusting enemy spawn rates and player growth rate accordingly.
- **public void Medium()**:
  Sets the game difficulty to "Medium," balancing enemy spawn rates and player growth.
- **public void Hard()**:
  Sets the game difficulty to "Hard," increasing enemy spawn rates and the player’s growth rate.
- **public bool Collision(GamePiece a, GamePiece b)**:
  Determines whether two game pieces are colliding by comparing their positions on the board.
- **public void GameEndDialog()**:
  Displays the end-game dialog, showing the player's score and giving options to retry or quit.
- **public void InitializeAll()**:
  Initializes or resets the game pieces and their corresponding collections for the game.
- **public void StopAllTimers()**:
  Stops any active timers related to fruit or game mechanics.

CLASS DIAGRAM:

```
+-------------------+
|     GamePiece     |
+-------------------+
| - objectMargins: Thickness
| - onScreen: Image
| - rotate: RotateTransform
| + Location: Thickness
| + Image: Image
| + Points: int
| + Size: int
| - random: Random (static)
+-------------------+
| + GamePiece(Image, int)
| + Move(VirtualKey): bool
| + CompareTo(GamePiece): int
| + Equals(GamePiece): bool
| + NewLocation(Thickness): void
+-------------------+

+-------------------+
|     MainPage      |
+-------------------+
| - nM: NotMain
| + placement: Random
| + practiceScrollViewer: ScrollViewer
+-------------------+
| + MainPage()
| + PieceUniqueSpot(string, double, List<GamePiece>): GamePiece
| + ResetGame(): void
| + GameTimerEffet(object, object): void
| + CreatePiece(string, double, int, int): GamePiece
| + FruitTimerEffect(object, object): void
| + CollideDots(): void
+-------------------+
+------------------+
|      NotMain      |
+------------------+
| + screenHeight: double
| + screenWidth: double
| + sizeOfTargets: double
| + sizeFruits: double
| + sizePlayer: double
| + score: int
| + gameOn: bool
| + currentMode: GameMode
| + gamePieces: List<GamePiece>
| + manyDots: List<GamePiece>
| + manyEnemies: List<GamePiece>
| + manyFruits: List<GamePiece>
| + enableEnlarger: bool
| + sizeEnlarger: double
| + enableEnemy: bool
| + enemySpawnRate: int
| + addedDotCount: int
| + limitDots: bool
+------------------+
| + NotMain(MainPage)
| + ShowDialogs(): Task
| + SpotTaken(double, double, double, List<GamePiece>): bool
| + StopAllTimers(): void
| + GameEndDialog(): void
| + Collision(GamePiece, GamePiece): bool
+------------------+
```

4. Change Log since previous submission of Lab 1A.

Date	Changes Made
2024-10-04	Added levels from Practice to Level 2
2024-10-08	Added Level 3 as a time trial
2024-10-10	Implemented score tracking, saving and displaying.
2024-10-14	Introduced challenge levels and dialog boxes for all the options for a player to choose. Inserted guide.
2024-10-16	Fixed bugs related to needing to click multiple times on dialog options for close or okay.
2024-10-17	Separated 2 classes from Main class and changed the documentation

5. Bug Report

Bug ID	Description	Status	Resolution
001	Needing to click multiple times on dialog options for close or okay.	Resolved	Updated the collision logic in NextStepDialog Method.
002	When a player at the beginning of the game chooses “No to all” and then chooses to practise, the option for challenge level remains at default. ie.Easy	Unresolved	Investigating the logic of the NextStepDialog Method.

References:

ChatGpt for checking overall structure, spelling and grammar of the documentation.