Flame Game: A Better Flappy Bird

Good Luck!

Complete the flutter project, using flame, a flutter game engine package to rebuild a flappy bird game, with procedural generation. Unlike the original game, the game will follow a dedicated story to your liking.

It's about the protagonist travelling the world you built to find something. It can be a damsel in distress, the antagonist or some awesome final item.

Technical requirements:

1. Use Flutter SDK version 3.0.0.
2. Use the latest version of Flame.

Game requirement:

1. The game can be broken into three rounds with each round completed, a small visual novel is played.
2. The game can be paused, resumed, restarted and quit.
3. The fundamental components of a Flappy Bird game are random pillars and annoying birds. Include the same principle of randomized walls and ever-moving playable characters to create your adaptation of the game.
4. Random walls can use Perlin noise or basic random function, based on your use case.
5. Add background audio, audio for each animation and for game objects that require it.
6. Utilize sprites and provide Sprite animations to show the character's current state and state change. Eg: Running to Jumping to Dying.
7. Calculates the score for the game. The scoring system can be designed to your liking and the round cut-off point can also be designed to your liking. Eg: When you reach a certain score the game is over, when you reach a certain score within a given time etc.

Note: These specifications only provide the requirements to develop a basic Flappy Bird game. Get creative, utilize the components of the game and adapt it to your liking. E.g.: Add traps and enable playable agents to shoot, develop enemies etc.

UI requirements:

1. The game will have a home page, which will consist of four buttons: Play, Score, Developer information and Quit.
2. The play button will trigger the game to instantiate.
3. The score button will navigate to a new screen, showing time, score and other necessary attributes related to the game. Use SQLite database to store the relevant information.
4. Developer information will navigate to a screen providing information about you, the developer.
5. Quit is Quit. It closes the application.

Note: Get creative with the UI.

Document requirements:

- Pre-Development Documentation:
    1. Use case diagram
    2. User stories Document
    3. Project Timeline Document

- Development Documentation:
    1. System Architecture
    2. Technical Specifications Document

- Post-Development Documentation:
    1. System Documentation: Comprehensive documentation of the system, including user manuals, administrator guides, and API documentation, if required.

Note: If you are wondering what the difference between the **Technical Specifications Document (TSD)** and **System Documentation (SD)** is. TSD is meant for the development team and SD is meant for a wider audience: developers, testers, end-users, system administrators, and other stakeholders.
System Architecture is a document that goes over the higher-level architecture of the project.

Coding Standards:

1. Use Flutter BLoC for state management.
2. Use const Constructor wherever it is needed.

3. Use naming conventions:
    - Use snake_case (lowercase with underscores) for libraries, directories, packages, and source files.
    - Start private variable names with underscores.
    - Use lowerCamelCase for constants, variables, parameters, and named parameters.
    - Use UpperCamelCase for classes, types, extension names, and enums.
    - Always use clear and meaningful names to improve code readability.

4. Utilize other Flutter coding standards.
5. Always provide comments for functionality.

Section Important takeaway:

1. Null safety
2. Stateful widget/ Stateless widget.

1. Collision Detection
2. Procedural Generation
3. Perlin noise
4. Flutter BLoC State management
5. Sprite/ Sprite Animation
6. SQLite database.

References

1. Flutter best practices: https://www.intelivita.com/blog/flutter-development-best-practices/
2. Flame Documentation: https://docs.flame-engine.org/latest/
3. Flame pub dev: https://pub.dev/packages/flame