**UML Diagrams for the project “Dino Run”**

**Game.h**

----------------------------------------------

| Game

----------------------------------------------

| - gameOver: bool

| - pause: bool

| - score: int

| - hiScore: int

| - dino: Dino

| - morning: NightnMorning

| - night: NightnMorning

| - trees: Trees[MAX\_TREES \* 2]

| - treesPos: Vector2[MAX\_TREES]

| - treeSpeedX: float

| - superfx: bool

| - win: bool

| - background\_music: MusicPlayer

| - gameOverSound: Sound

| - winSound: Sound

| - jumpSound: Sound

| - scoreSound: Sound

| - emotionalSound: Sound

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| + Game()

| + gameLoop(): void

| + InitGame(): void

| + UpdateGame(): void

| + DrawGame(): void

| + UnloadGame(): void

| + UpdateDrawFrame(): void

| + restartGame(): void

----------------------------------------------

----------------------------------------------

| Dino

--------------------------------------------

|+ position: Vector2

|+ width: int

|+ height: int

|+ color: Color

|+ texture: Texture2D

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

----------------------------------------------

| NightnMorning

----------------------------------------------

| + position: Vector2

| + width: int

| + height: int

| + color: Color

| + texture: Texture2D

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

----------------------------------------------

| Trees

----------------------------------------------

| + rec: Rectangle

| + color: Color

| + active: bool

| + texture: Texture2D

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

----------------------------------------------

| MusicPlayer

----------------------------------------------

| + music: Music

| + playing: bool

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

In this diagram, the ***Game*** class represents the main game logic. It contains various properties such as ***gameOver***, ***score***, ***dino***, ***trees***, and ***background***\_***music***. The class also defines methods for initializing the game, updating, and drawing the game state, unloading resources, and restarting the game.

The `Dino` class represents the player character. It has properties like `position`, `width`, `height`, `color`, and `texture` that define its appearance and position in the game.

The `NightnMorning` class represents the morning and night entities in the game. It has similar properties as the `Dino` class.

The `Trees` class represents the trees in the game. It has properties like `rec`, `color`, `active`, and `texture` that define the appearance and state of the trees.

The `MusicPlayer` class represents the music player in the game. It has properties like `music` and `playing` to control the background music.

Note that the diagram represents the structure and relationships between classes and their members, but it does not include the specific details of member methods and variables.

**Utils.h**

----------------------------------------------

| Utils

----------------------------------------------

| + DrawCentralizedText(text: string,

| fontSize: int, color: Color,

| width: int, height: int): void

| + DrawButton(text: string, fontSize: int,

| color: Color, width: int, height: int,

| x: int, y: int, callback: void (\*)()):

| void

---------------------------------------------

```

In this updated diagram, a new class called `Utils` is added to represent the utility functions. It contains two methods: `DrawCentralizedText` and `DrawButton`, which are responsible for drawing centered text and buttons, respectively.

`Utils` class is not associated with the `Game` class or other existing classes, as it contains standalone utility functions.

Main.cpp

In main, a new class called `Utils` is added to represent the utility functions. It contains two methods: `DrawCentralizedText` and `DrawButton`, which are responsible for drawing centered text and buttons, respectively.

Note that the `Utils` class is not associated with the `Game` class or other existing classes, as it contains standalone utility functions.

**Game.cpp outline**

A picture containing line, screenshot, diagram, plot

Description automatically generated

This diagram shows the game starting with initialization, then entering the game loop. In each loop, the game updates (including checking collisions, moving trees, handling jumps, and updating the score) and then draws the game elements (background, dino, trees, score, pause message, and game over message). This continues until the game ends.

**Building the game**

**Makefile**

default: g++ ../source/\*.cpp -o Game.exe -O2 -Wall -Wno-missing-braces -I ../include/ -L ../lib/ -lraylib -lopengl32 -lgdi32 -lwinmm

This is a Makefile for a project using the Raylib library. Makefiles are used to automate the build process of a project. This Makefile is set up to build a project using the g++ compiler on a Windows system.

Here's a breakdown of the command:

- `g++`: This is the GNU C++ compiler. It's used to compile C++ code.

- `../source/\*.cpp`: This tells the compiler to compile all .cpp files in the `../source/` directory.

- `-o Game.exe`: This option specifies the output file name. In this case, the output executable will be named `Game.exe`.

- `-O2`: This option tells the compiler to optimize the code for speed. There are several levels of optimization, with `-O2` being a moderate level that increases performance without significantly increasing compile time.

- `-Wall`: This option enables all compiler's warning messages. This is useful for catching potential issues in the code.

- `-Wno-missing-braces`: This option disables the warning about missing braces. This can be useful if your code uses initializers that don't require braces.

- `-I ../include/`: This option tells the compiler where to look for header files. In this case, it's looking in the `../include/` directory.

- `-L ../lib/`: This option tells the compiler where to look for library files. In this case, it's looking in the `../lib/` directory.

- `-lraylib -lopengl32 -lgdi32 -lwinmm`: These options link the specified libraries to the project. In this case, it's linking the Raylib library, the OpenGL library, the GDI library, and the Windows Multimedia library.

Raylib is a simple and easy-to-use library to enjoy videogames programming. It's highly portable and easy to learn, making it a great choice for beginners to game development. You can find more information about Raylib in the official documentation - <https://www.raylib.com/cheatsheet/cheatsheet.html>