**Project Description:**

Title: The Ultimate Golazo

Overview: The Ultimate Golazo is a bird's-eye view soccer game inspired by EA FC 24, it will be created using standard Python and the CMU Graphics library. The game simplifies the dynamics of soccer into an engaging experience, focusing on fundamental elements like tackling, passing, shooting, dribbling, and goalkeeping. Designed to be intuitive and user-friendly, it aims to offer a fun and strategic soccer simulation while showcasing the capabilities of Python programming in game development.

**Similar Projects:**

Drift112 is a key inspiration for my project, The Ultimate Golazo. I'm particularly impressed by how smoothly the cars move in Drift112, a feature I plan to incorporate into my soccer game. By using 2-D lists, I aim to achieve similar fluidity in player movements, ensuring a more realistic and enjoyable gameplay experience.

Another project I've looked at is New Star Soccer. It has engaging shooting mechanics. While I appreciate this aspect, in The Ultimate Golazo, players will have direct control over shooting, offering a more interactive and dynamic experience. This change not only adds a strategic depth to the game but also allows players to feel more involved in the action.

**Structural Plan:**

My project, The Ultimate Golazo, will be organized through OOP, using classes and functions to create an engaging soccer simulation game.

Classes:

Player Class: Manages player attributes like shooting, passing, and defending, rated from 1-100. Methods include actions like dribbling and shooting.

Game Class: Controls game mechanics such as scorekeeping, time management, and player positions.

Ball Class: Handles the ball's movement and interactions with players.

Modules:

Graphics Module: Used for 2-D visualization of the field, players, and ball.

Random Module: For generating random events within the game.

Math Module: For calculations related to player and ball movement.

Key Functions:

Tackle Function: Determines the outcome of tackles using player attributes and proximity.

Foul Detection: Detects fouls based on player interactions, following soccer rules.

Player Movement: Players move using a grid-based system, allowing for actions like passing, shooting, and tackling.

Gameplay Mechanics:

Phases of Play: The game includes phases like kickoff, general play, and set-pieces, each with distinct rules and player behavior.

**Algorithmic Plan:**

The algorithmically challenging part of The Ultimate Golazo involves creating realistic player movement and ball control within the 2-D grid system. This includes simulating aspects like dribbling, passing, and shooting in a way that feels natural and responsive.

Key Challenges and Implementation:

Player Movement Algorithm: The movement of players on the field will be controlled by algorithms that consider their speed, agility, and current action (like dribbling or running towards the ball). This will involve complex pathfinding algorithms to navigate players around opponents and towards the ball.

Ball Control Mechanics: Implementing a system for ball control, which includes passing and shooting. The challenge here lies in accurately calculating ball trajectory, speed, and interaction with players. This will involve physics-based algorithms to simulate realistic ball behavior.

Collision Detection and Response: A crucial part of the game is determining collisions between players and between players and the ball. This will require efficient algorithms to detect collisions and determine the appropriate response, whether it's a change in ball direction, a tackle, or a foul.

By focusing on these algorithmic challenges, The Ultimate Golazo will aim to provide a realistic and engaging soccer experience, with smooth player movements and responsive ball control that mirrors real-life soccer dynamics.

**Timeline Plan:**

Week 1 (November 20 - November 26): Initial Development

Setup the basic game environment using CMU Graphics.

Develop the Player Class with basic attributes (shooting, passing, etc.).

Create a simple representation of the soccer field.

Week 2 (November 27 - December 3): Core Mechanics Development

Implement the Ball Class and basic ball movement mechanics.

Develop initial player movement logic and controls.

Begin work on the Game Class to manage scores and game states.

Week 3 (December 4 - December 6): Finalization and Polishing

Finalize collision detection for tackles and implement foul detection logic.

Optimize player and ball movements for smoothness.

Conduct thorough testing and debugging.

Prepare for TP3 submission, including documentation and demo video.

**Version Control Plan:**

For my project, "The Ultimate Golazo," I am implementing a straightforward yet effective version control system using a combination of local storage on my Mac and iCloud for backups. This system allows me to manually manage versions of my project and ensures that my work is securely backed up in the cloud.

Key Elements of the Plan:

Local Version Control: I maintain a dedicated project folder on my Mac where I develop the game. Inside this folder, I create subfolders labeled with dates or version numbers for each significant development stage, ensuring that I can track changes and revert to previous versions if necessary.

Cloud Backup: In addition to local storage, I regularly upload the entire project folder to iCloud. This serves as a secure offsite backup, protecting against data loss due to hardware issues or accidental deletion.

Backup Strategy:

Each significant update or completed feature leads to the creation of a new subfolder in my local project directory.

After every coding session, or at least daily, I upload the latest version of the project folder to iCloud, maintaining an up-to-date cloud backup.

This would be how I would keep it on my desktop, and I will back this up to my iCloud daily.

A screenshot of a computer

Description automatically generated

**Module List for The Ultimate Golazo:**

My project will primarily use standard Python libraries and the CMU Graphics library, which is a requirement of the assignment. Here is the list of modules and technologies I plan to use:

Standard Python Libraries: These include built-in Python libraries like random for generating random events in the game and math for calculations related to movement and ball physics.

CMU Graphics Library: This is the main graphics library provided by the course for creating graphical user interfaces and animations. It will be used to render the game's visual elements, including the soccer field, players, and ball.

iCloud: Although not a Python module, iCloud will be used as an external technology for backing up the project files. This ensures data safety and allows for recovery in case of local data loss.

I do not plan to use any additional external modules or technologies that would require approval through a tech demo. The project will strictly adhere to the course's guidelines by focusing on the core Python programming language and the CMU Graphics library for all development and design aspects of the game.

**TP 1 Update:**

I don’t have any changes for the design aspect in terms of the storyboard and my vision. But in terms of my code, I completed the initial coding phase, focusing on basic game mechanics like team and player setup, ball movement, and game state management. I faced challenges in creating realistic ball movement but addressed this by refining the trajectory calculations and mouse-based controls. I also worked on smoothing out game state transitions for a better player experience. Next, I plan on adding AI for opponent teams and additional gameplay features like scoring and a game timer.

**TP2 Update:**

I changed what happens for out of bounds. To make the game flow better I made it so that the ball can’t be out of bounds and instead it behaves like volta where if it hits the out line, it bounces right back. That’s really all I changed; all I have left to do is to make my AI opponent better.

**TP3 Update:**

I added an AI that plays for the opponent’s team. It defends and attacks. I also made the goalkeeper in my team play by itself. I have a better physics engine. The ball gets slower over time after its passed and if it hits a player, it’s stopped. It’s not stopped if it hits a player within a second of it being passed. This is to make sure it doesn’t stop when it hits itself after the pass.