TDD (Technical Design Document)

GeeksInstitute - LaStartupStation

Project Name: Tbourida: The Noble Charge

Name	Role
Abdellah Aoukrad	Game Dev / Unity / Producer / Game Designer
Zakaria Rezki	Graphic Designer / 3D / 2D
Soufiane Mjadi	Graphic Designer / 3D / 2D
Yassine Ait HMAD	Video/Image Editor / Documentation / Marketing

Version Table

Version No.	Date	Author(s)	Description
1.0	2025-12-18	Yassine Ait Hmad	Draft TDD with same sections as the game idea docs.
2.0	2025-12-23	Abdellah Aoukrad	Final TDD with proper TDD sections.

1. Game Presentation

Tbourida: The Noble Charge is a 3D rhythm-simulation game featuring a stylized low-poly aesthetic. The technical scope focuses on a flocking algorithm for the horse formation (Sorba) and a deterministic timing system for the shooting phase. By utilizing low-poly assets, the project ensures high visual clarity and optimal performance across mobile and PC platforms, allowing for a higher number of on-screen agents without performance degradation.

2. Platforms and Hardware Specifications

- **Target Platforms**: PC (Windows 10/11) and Mobile (Android API 26+, iOS 13+).
- **Performance Advantage**: The move to a low-poly style significantly lowers the minimum hardware requirements. This allows for smooth 60 FPS gameplay on entry-level mobile devices, which is critical for the accuracy of rhythm-based mechanics.

3. Development Environment

- **Game Engine**: **Unity 6.3 LTS**. The engine is selected for its efficient Universal Render Pipeline (URP), which is ideal for stylized rendering and flat-shaded low-poly aesthetics.
- **3D Modeling**: Blender, utilizing an "un-smoothed" workflow to preserve hard edges and faceted surfaces.
- **Version Control**: Git (GitHub).

4. Technical Risks

- **Visual Readability**: Small low-poly details might become "noisy" at high speeds.
  - *Solution*: Use high-contrast vertex colors and avoid detailed textures.
- **Animation Deformation**: Low-poly meshes can "pinch" at joints during high-speed gallop cycles.
  - *Solution*: Implement stylized, snappy animations with linear interpolation and weighted bone influences to maintain the "faceted" look even during movement.

5. Production Pipeline

- **Data Production**: 3D meshes are produced with a low polygon budget (e.g., <2,000 triangles per horse).
- **Export/Import**: `.FBX` files are exported with "Flat Shading" settings. Normal data is ignored in favor of faceted face normals.
- **Coloring**: Vertex colors are used extensively to define character and horse patterns, reducing the need for texture memory and Draw Calls.

6. Tools to Develop

1. **Arena Designer**: A custom Unity tool to allow artists to design a Tbourida arena inside Unity, place the Moroccan objects/environment directly in Unity for each arena and export the placement data to re-create the arena in runtime without having hardcoded maps/arena for each place.
2. **Sorba Tuner**: A ScriptableObject-based tool to adjust the "spacing" and "cohesion" values of the AI horses in real-time.

7. Software Architecture

- **Top-Down Architecture**: Managed by a central `GameManager`.
- **Key Modules**:
  - `InputHandler`: Maps touch and keyboard inputs to horse movement.
  - `FormationAI`: Manages the local offsets of the 10+ AI agents relative to the player.
  - `RhythmEngine`: The core logic that calculates the delta-time between the player's trigger and the "Green Line."

8. Graphic Rendering Techniques & Complexity

- **Render Pipeline**: **URP (Universal Render Pipeline)**.
- **Shaders**: Custom Toon Shaders or Flat-Lit shaders that emphasize the edges of the polygons.
- **Post-Process Stack**: Bloom and Color Grading (LUTs) are used to recreate the warm, dusty "Moussem" atmosphere without complex lighting.
- **Scene Complexity**: Low polycounts enable a high "Crowd" density in the stands using GPU Instancing.

9. Performances

- **Target**: Stable **60 FPS** is the baseline.
- **Optimization**: Because the models are low-poly, the physics engine can use simplified box colliders for all 10+ horses, drastically reducing CPU overhead during the "Drifting" phase.

10. Technical Implementation of Gameplay Mechanics

- **The Drift Logic**: A Perlin Noise function generates a lateral force value. The player’s input directly counters this value in the `FixedUpdate` physics loop.
- **Synchronized Shot**: Upon the player's input, the `RhythmEngine` captures the `Time.time`. AI agents are programmed to fire with a randomized delay between 10ms and 50ms to simulate natural human variation.
- **Scoring Math**: The score is a result of a standard deviation calculation of all shot timestamps within the Sorba.

11. Menus & HUD

- **Technique**: Flat UI (uGUI) with sharp edges to match the 3D aesthetic.
- **World-Space Integration**: The shooting lines (Red/Green) are meshes placed slightly above the ground plane to avoid Z-fighting.

12. Audio

- **Technique**: Audio triggers are synced with the visual muzzle flashes.
- **Dynamic Music**: The pitch and volume of the traditional percussion track are mapped to the current velocity of the horse.

13. Code Writing Conventions

- Standard C# conventions: `PascalCase` for methods/classes, `camelCase` for local variables.
- Mandatory use of `[SerializeField]` for variables that need to be tuned by designers in the low-poly environment.

14. Equipe et organisation

- **Workflow**: Artists deliver low-poly `.fbx` files; developers integrate them into Prefabs.
- **Coordination**: Weekly builds are tested on target mobile hardware to ensure the low-poly style translates well to small screens.