Mid-term Written Report: Ultimate Chicken Horse 3D

Topic 2: Interactive Game

An Yan
yana24@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Zhenbang Pan
pzb24@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

ABSTRACT

This report summarizes the progress of our project "Ultimate Chicken Horse 3D", a web-based multiplayer party platformer game. We detail the project goals, technical points, finished aspects including the toon shading rendering pipeline and physics integration, and the plan for remaining tasks such as networking and character animation.

1 PROJECT GOAL

We aim to develop a web-based, 3D multiplayer party platformer game inspired by *Ultimate Chicken Horse*. The core gameplay loop involves two distinct phases: a **Building Phase**, where players strategically place traps and platforms to hinder opponents, and a **Running Phase**, where players compete to reach the goal using the built environment. The game features a stylized cartoon aesthetic and utilizes a P2P network architecture for online multiplayer interaction.

2 TECHNICAL POINTS

Beyond the basic requirements , we plan to implement the following advanced features:

- **An Yan:**
  - **Online multiplayer system (4pts):** Implementing a Host-Authoritative P2P architecture using WebRTC.
  - **Customizable character appearance (2pts):** Allowing players to select different animal skins using texture atlas UV mapping.
  - **Easy installation or online access (2pts):** Deploying via GitHub Pages with automated CI/CD.
  - **Environment lighting (1pts):** Implementing Toon-style lighting with rim lights and shadow maps.
- **Zhenbang Pan:**
  - **Articulated animals with rigging and skinning (3 pts):** Creating a repeatable rigging/skin workflow for animal characters with blender, Mixamo and Three.js.
  - **Surface interaction physics (2 pts):** Implementing gameplay surfaces with distinct contact properties, e.g. honey/ice.
  - **Additional auxiliary interfaces in 3D (2 pts):** Designing in-world 3D roll/tool selection interfaces highlighting intuitiveness and interactivity.
  - **User-friendly layout with visually appealing design (1 pt):** Implementing UI components embedded in 3D that provide consistent style, immersive experience and smooth camera transition.
  - **Synchronized audio (1 pt):** Event-driven audio system (BGM + SFX) with positional audio for key events (e.g. trap trigger, jump, goal).

3 EXTERNAL TOOLS

- **Rendering & Engine:** Three.js (WebGL), Vite (Build Tool).
- **Physics Simulation:** Cannon-es (Rigid body physics).
- **Networking:** PeerJS (WebRTC wrapper for P2P connection).
- **Modeling & Animation:** Nano Banana Pro (text-to-image generation) for concept images, and the image-to-3D pipeline from Xiang et al.[1] to convert those images into 3D models; Blender (cleanup, UV mapping, harden normals) for final asset preparation, and Mixamo (auto-rigging) when appropriate.
- **Deployment:** GitHub Actions (CI/CD).

4 FINISHED TECHNICAL ASPECTS

We have established the core technical framework and art pipeline.

- **Rendering Pipeline (Cartoon Style):**
  - We rejected standard PBR rendering in favor of a stylized **Toon Shader**. We implemented `MeshToonMaterial` with a custom 16-step Gradient Map to achieve distinct light banding.
  - Resolved "pillow shading" artifacts on low-poly models by using **Harden Normals** in Blender and correct export settings.
- **Asset Pipeline:**
  - Models for gameplay tools (platforms, spikes, crossbows) and character roster (chicken, horse, sheep, monkey, raccoon, lizard, rabbit) were produced using the modeling pipeline described in the "External Tools" section (Nano Banana Pro $\longrightarrow$ Structured 3D Latents image$\longrightarrow$3D pipeline $\longrightarrow$ Blender cleanup/UV $\longrightarrow$ optional Mixamo rigging).
- **Physics Engine Integration & Gameplay Items:**
  - Integrated **Cannon-es** with Three.js. Implemented a sync mechanism where the physics body drives the visual mesh.
  - Implemented diverse gameplay items with specific physics logic, focusing on **Traps**: Implemented *Crossbows* with automatic firing timers and object-pooled projectiles, and *Spikes* as static hazards.
- **3D in-world UI:**
  - Implemented UI components as **Three.js Scene Objects** (menu, roll selection, tool selection).
  - Implemented **Smooth Camera Transition** to shift the player's focus naturally and immersively between in-world UI elements, such as menus, tool selection, and the build/run mode.
- **Networking Infrastructure:**
  - Established the core P2P communication architecture using **PeerJS**.

  - Implemented connection lifecycle management (host/client handshake) and a packet-based communication protocol supporting both unicast and broadcast.
- **Infrastructure:**
  - Set up the **GitHub Actions** workflow. The project is automatically built and deployed to `github.io` upon pushing to the main branch, solving path resolving issues (base URL configuration).

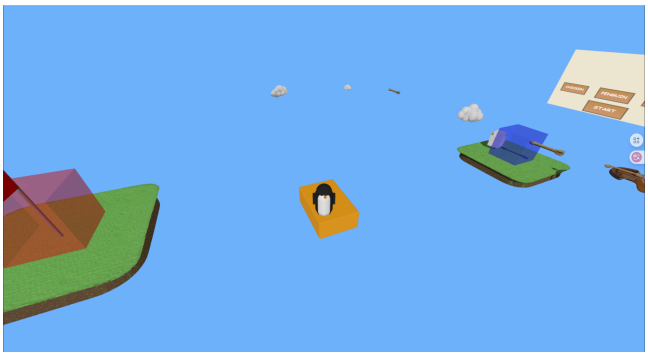

Figure 1: Current game scene

5 PLAN FOR REMAINING TECHNICAL TASKS

- **Networking (Gameplay Sync):** Building upon the established P2P infrastructure, finalize the state synchronization for the "Building Phase" (syncing object placement) and "Running Phase" (interpolating player positions to handle latency).
- **Character Animation:** Rig and animate models produced by the External Tools pipeline (Mixamo or manual rigging), and implement the `AnimationMixer` state machine to blend Idle, Run and Jump.
- **Interactive Items & Surfaces:** Implement physics logic for "Honey" (high friction) and "Ice" (zero friction). Implement interactive objects including *Springs* (high restitution), *Conveyors* (constant force), and *Coins* (collection logic).
- **Game Loop Logic:** Implement the turn-based logic, scoring system, and the "Replay System" (recording state snapshots for playback).
- **Audio & Polish:** Add spatial sound effects and refine the UI interactions.

6 DETAILED SCHEDULE

- **Nov 15 - Nov 30:**
  - Project initialization, tech stack selection (Three.js + Cannon-es), and setting up the CI/CD pipeline. (Completed)
  - Building up core gameplay prototype, implementing the "Building Phase" (Raycasting for object placement) and basic "Running Phase" physics. (Completed)
- **Dec 1 - Dec 14:**
  - Implementing in-world 3D UI and art style validation. (Completed)
  - Implementing PeerJS data channels for syncing game states and player inputs.
  - Importing rigged characters, setting up the animation state machine, and implementing character customization.
- **Dec 15 - Dec 31:**
  - Gameplay refinement, adding interactive items (Springs, Conveyors, Coins) and fluid/ice mechanics.
  - Implementing audio, scoring logic, and the replay system.
  - UI beautification, bug fixing, and preparation for the In-class Presentation.
- **Jan 1 - Jan 14:**
  - Final Report writing and code cleanup.

REFERENCES

[1] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. 2025. Structured 3D Latents for Scalable and Versatile 3D Generation. arXiv:2412.01506 [cs.CV] https://arxiv.org/abs/2412.01506

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009