University of the Balearic Islands
Bachelor's Degree in Computer Engineering

21714 — Computer Graphics · 2025–2026 Academic Year

Final Report

Cinematic of a Boxing Match in the UIB Arena

Baki Hanma vs Yuji Itadori

Team members: Andreu Massanet
Diego Malagrida

Date: June 2026

Contents

1 Introduction

This document describes the final project of the course 21714 Computer Graphics, 2025–2026, at the UIB. The deliverable is a cinematic (not a game) of roughly one minute that recreates a boxing match between Baki Hanma and Yuji Itadori in a UIB-inspired arena/stadium. The report is organized by **development stages**, as required by the assignment brief: each stage has its own section and is closed with a final report (user manual, advantages and drawbacks, critical assessment, conclusions and appendices).

1.1 Objectives of the project

The project aims to integrate into a single audiovisual production the techniques covered in the course: modeling of the set and props, character and camera animation, materials and textures, cinematic lighting, an interaction layer and advanced realism techniques (particles, physics, volumetrics and emissive fire).

1.2 General description of the 3D scene

The scene was built in Blender 4.5 with the Cycles render engine (EEVEE preview). The cinematic is resolved as a single continuous take chaining 9 cameras, switched via timeline markers, covering the opening on the sign, the crowd, the descent to the ring, the introduction of both boxers, three shots of the fight and the victory with the trophy. The sequence spans frames 1 to 2176 at 30 fps, i.e. about 72 seconds, at a resolution of 1920×1080.

Figure 1: Overview of the UIB arena and the ring where the fight takes place.

2 Stage 0: Storyboard

The storyboard is the upfront visual planning of the cinematic. Its purpose is to establish, before animating and rendering, the sequence of shots, the composition of each frame and the narrative rhythm of the fight, serving as a guide for assembling the nine cameras switched by timeline markers.

2.1 Overall goal of the scene

- ❒ **What does it depict?** A ∼1-minute cinematic of a boxing match in the UIB arena: a ring lit by concert spotlights at the center of a stadium with stands, a crowd and a big screen.
- ❒ **Theme.** Sports spectacle / boxing night. Two recognizable fighters face off, **Baki Hanma** and **Yuji Itadori**.
- ❒ **What we want to show.** A complete narrative arc resolved as a **continuous long take of chained cameras**: from the presentation of the arena to the finale (bell, exchange of punches, KO and the winner's celebration with pyrotechnics, confetti and trophy). The technical goal is to integrate modeling, animation, materials and textures, cinematic lighting, keyboard interaction and realism techniques (particles, physics and volumetrics) into a single scene.

2.2 Concept definition

- ❒ **What does the scene depict?** A boxing night in the UIB arena. The focus is on the central ring; the stadium, the stands and the big screen provide context and scale. The action goes from the spectacular opening to the KO and the victory.
- ❒ **Which articulated objects will appear?**
  - **Baki** and **Itadori**: meshes with a full humanoid skeleton, deformed by *skinning*. Baki wears a bare torso and shorts; Itadori, a casual outfit (shirt, trousers, shoes and an accessory).
  - **Crowd**: a base character with a cyclic-action rig, instanced across the stands.
  - **Stage spotlights**: an articulated moving head on its base.
  - **Confetti cannons** with their particle emitter.
- ❒ **What keyboard interaction will the user have?** Two custom **consoles** (Python modal operators): a **light** one, which turns the scene's lighting groups on and off live, and a **camera** one, which jumps between the nine shots; plus the cinematic's playback control (see Section 2.7).

2.3 Initial idea: the six panels

The storyboard is condensed into six illustrated panels (Figure 2) that structure the roughly 72 seconds of the piece:

1. **Arena opening**: an orbit around the sign and a wide view of the UIB stadium that introduces the set; the screen shows the "BAKI vs ITADORI" card.
2. **Baki's introduction**: a shot of the winning fighter on the ring.
3. **Itadori's introduction**: a shot of the rival, who will end up losing.
4. **Exchange of punches**: the fight itself, covered with wide, low-angle and close shots.
5. **KO**: Itadori's defeat (dejected posture, "sad idle").
6. **Victory**: Baki triumphant next to the trophy, with confetti.

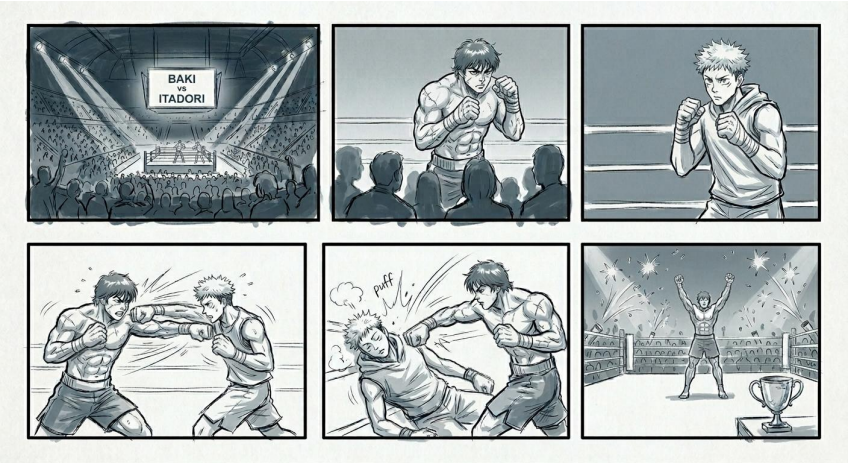

Figure 2: Illustrated storyboard of the cinematic: the six panels covering the complete narrative, from the arena opening to the victory.

2.4 Mini storyboard: camera, lighting and materials

For each panel we note the camera, the lighting, the dominant materials and the planned interaction.

#	**Description / movement**	**Camera**	**Lighting**	**Materials**
1	Arena opening: the stadium lights up, the spotlights sweep the stands and the screen shows the "BAKI vs ITADORI" card.	Wide shot, slow orbit around the sign.	Colored concert spotlights + screen glow.	Stands, structure, emissive screen.
2	Baki's introduction: he steps into the ring and raises his fists in guard.	Medium shot of the fighter.	Ring key light + *rim*.	Skin, shorts, canvas with UIB logo.
3	Itadori's introduction: his rival, on the other side of the ring, ready.	Medium reverse shot.	Key light + ring *rim*.	Casual outfit.
4	Exchange of punches: the bell rings, both trade blows at the center.	Low, dynamic shot.	Ring *reveal* + key light.	Skin, outfits, canvas.
5	Decisive blow (KO): Itadori falls to the canvas.	Close shot of the impact.	High-contrast key light.	Skin, outfits, canvas.
6	Victory: Baki raises his arms; confetti cannons and trophy.	Wide shot of the ring.	Pyrotechnics + crowd *wash* + spotlights.	Confetti, metallic trophy, canvas.

Planned interactions during playback: the user can turn any light group on/off with the console (e.g. turn off the concert spotlights in panel 1), pause the cinematic and jump between shots with the camera console.

2.5 Models and articulation hierarchy

❒ **List of objects to model:**

- **Baki:** a mesh with a full humanoid skeleton (rig `Armature`); he is the winner.
- **Itadori:** an outfit in four separate meshes (shirt, trousers, shoes and accessory) weighted to the full skeleton (rig `Bip01`); he loses by KO.
- **Ring and UIB arena:** platform, canvas, ropes and posts, stands (*lower / upper bowl*), structure, entrances and screen.
- **Crowd:** a base character in three levels of detail (LOD) to optimize rendering of the crowd.
- **Props:** the trophy with its table, confetti cannons, articulated stage spotlights and the decal plane with the UIB logo on the canvas.

❒ **Articulation hierarchy (example, a fighter):**

`Empty (scale)` `Armature` `Mesh` (skinning deformation)
`Hips` `Spine` `Chest` `Shoulder` `Arm` ... (hands)
`Neck` `Head`
`Hips` `Leg` `Knee` `Ankle` `Toe`

`Stage spotlight:` `Root` Head (moving head) `Beam / Lens / Body`

`Confetti cannon:` `Prop_ConfettiCannon + Emitter` (2,400-particle system)

The characters deform by **skinning** (vertex weights and an *Armature* modifier), not by parenting props to bones: Baki's mesh is a single mesh and Itadori's outfit is four meshes weighted to the same skeleton. The **spotlights** orient their head (`Head`) from the base (`Root`); the **cameras** (nine shots) are switched via timeline markers bound to each camera.

2.6 Animation planning

A sequence of **nine shots** chained by timeline markers, frames 1–2176 at 30 fps (≈72 s). Times according to the actual markers:

Time	Shot / camera	What happens
0–10 s	1 · Opening (`Cam_01_-Sign`)	Orbit around the sign; the arena lights up, spotlights sweep the stands, screen shows the card.
10–20 s	2 · Crowd (`Cam_02_Crowd`)	A pass over the stands; the crowd cheers while seated.
20–32 s	3 · Descent (`Cam_03_-RingTop`)	The camera descends toward the ring.
32–38 s	4 · Baki (`Cam_04a`)	Introduction of Baki in guard.
38–45 s	5 · Itadori (`Cam_04b`)	Introduction of Itadori in his corner.
45–49 s	6 · Fight *wide* (`Cam_05a`)	Bell; exchange of punches, wide shot.
49–54 s	7 · Fight *low* (`Cam_05b`)	Exchange from a low angle.
54–64 s	8 · Fight *close* / KO (`Cam_05c`)	Close shot; decisive blow and Itadori's KO.
64–72 s	9 · Victory (`Cam_06_-Trophy`)	Baki raises his arms; confetti and trophy.

❒ **Characters:** Mixamo clips retargeted and combined into **NLA** tracks per fighter (introduction, *idle*/sway, *fist fight* and finale: *victory* for Baki, *sad idle* for Itadori).

- ❒ **Crowd:** cyclic actions (applause, cheering, disbelief) and a seated-to-standing transition in moments of tension.
- ❒ **Timed effects:** pyrotechnics at the entrance, continuous ambient smoke and, at the victory, the confetti cannons with a physics-driven fall.
- ❒ **Audio** (in the *Video Sequence Editor*): bell at the start and end, a layered crowd bed and a crowd roar at the KO.

2.7 Keyboard control

Since **Blender 4.x does not include the Game Engine**, interaction is handled with **custom Python modal operators** and the scene's playback control. There are two consoles, each with its panel in the "UIB" tab and its shortcut: the light one *modifies the scene* (turns lighting groups on and off in the viewport and render at once) and the camera one navigates between shots.

Shortcut / key	Action
`Ctrl+Shift+L`	**Light Console**: `[1]`–`[8]` toggle the 8 light groups (concert spotlights, pyrotechnics, *rim*, stage spotlights, key light, screen, crowd *wash*, ring *reveal*); `[0]` turns them all on.
`Ctrl+Shift+C`	**Camera Console**: `[1]`–`[9]` or the arrow keys jump between the nine shots.
`Esc`	Closes the active console.
`Space`	Plays / pauses the cinematic.

Node logic: additionally, the UIB logo decal on the canvas and the emissive stadium screen are built with material node graphs (visual logic, no code).

2.8 Storyboard validation

Checks before moving to production in Blender:

- ✔ **Objects defined:** fighters, ring, arena, crowd and props listed, with their articulations.
- ✔ **Animations defined:** NLA clips per character, cyclic crowd, 9 camera shots and timed effects, with an approximate timing of 72 s.
- ✔ **Interaction defined:** light and camera keyboard consoles and playback control, achievable without a Game Engine.
- ✔ **Cameras known:** a sequence of nine shots chained by markers.
- ✔ **Lighting planned:** concert spotlights, ring *rim* and *reveal*, key light, crowd *wash*, screen and pyrotechnics, controllable from the console.

3 Stage 1: Modeling

Modeling covers the arena, the two boxers, the scene props and the crowd, organized in the outliner within the numbered groups `01_SET`, `02_CHARACTERS` and `03_CROWD`.

3.1 The UIB arena

The set (`01_SET`) reproduces an indoor stadium. Its centerpiece is the *ring*, made up of the platform, the canvas, the ropes and the posts. Around it the stands rise in two rings, `LowerBowl` and `UpperBowl`, complemented by the seat mesh (`Bowl_Seats`) and the stand detailing (`Bowl_Detailing`). The shell is formed by `Structure`, the `Access` (entrances) and a big screen (`Screen`). The rest of the set dressing is provided by `Stage_Fixtures`, `Ringside_Dressing` and `Arena_Dressing`, along with the floor (`Floor`).

3.2 Characters

Two fighters were modeled (`02_CHARACTERS`):

- **Baki Hanma**, the winner: a single mesh with a full humanoid skeleton (rig `Armature`), with a bare torso and shorts. His materials are *Body*, *Clothing*, *Hair* and *Eyes* (Figure 3).
- **Yuji Itadori**, the loser (gets KO'd): a set of four casual-outfit meshes (shirt, trousers, shoes and an accessory, set `Male_Lobby_*_Cos`), driven by the `Bip01` rig.

3.3 Props and collection organization

The props were modeled separately (trophy and table, confetti cannons, articulated stage spotlights and the UIB-logo *decal* plane). The whole scene is organized in the outliner into eight numbered groups (`01_SET` ... `08_SOURCE_hidden`). Articulation is handled by *skinning* and by object hierarchy:

- **Characters (skinning):** the meshes deform via vertex weights and an *Armature* modifier (Baki is a single mesh; Itadori's outfit, four meshes weighted to the `Bip01` rig).
- **Stage spotlights:** the moving head hangs from the base via *empties*, which allows the beam to be aimed.
- **Confetti:** each emitter (`Confetti_Emitter_L/R`) sits next to its cannon and fires 2,400 particles.

3.4 Levels of detail (LOD)

Given the crowd's load (on the order of 16.7 million polygons), three levels of detail were used: a low-poly `nivel1` ($\sim$4,500 polygons) for the crowd, and two higher-complexity levels (one based on Itadori and another on Baki) reserved for foreground instances.

Figure 3: Model of Baki Hanma, the winning boxer, with his humanoid skeleton (rig `Armature`); he appears with a bare torso and shorts, and deforms by *skinning*.

4 Stage 2: Animation

The cinematic's animation combines character motion, camera switching, crowd behavior and a synchronized soundtrack, all orchestrated on the timeline of `final_scene.blend` (frames 1 to 2176 at 30 fps, ≈72 s).

4.1 Characters, keyframes and NLA tracks

Both boxers are animated from Mixamo clips. Each clip is retargeted to the target skeleton and, once adapted, distributed across tracks of the NLA (Non-Linear Animation) editor, a per-character organization that allows chaining, overlapping and adjusting the timing of each action without re-animating by hand. Baki Hanma (rig `Armature`) chains four actions: introduction, sway ("rockin"), fight ("fist fight") and "victory". Yuji Itadori (rig `Bip01`) follows a parallel progression — introduction, "rockin", "fist fight" — that culminates in a "sad idle", since he is KO'd and is the loser. The character meshes follow the motion by *skinning*, with no additional animation. Custom *keyframes* are reserved for the cut points between clips, the position adjustments of the characters on the ring and the shot-by-shot animation of cameras and lights.

4.2 Trajectories, imported objects and problems

The animation clips are **objects imported** from Mixamo in FBX format, kept in `blender-project/animations`. During retargeting a real problem arose: Itadori's arms clipped through the torso. It was fixed by applying the retarget in the rest pose. It should also be noted that the armatures hang from scaled and rotated *empties*, so the location channels (the translation trajectories) do not correspond 1:1 with the world, a fact that had to be taken into account when placing and moving the characters.

Figure 4: Shot of the fight between Baki Hanma and Yuji Itadori in the ring.

4.3 Camera sequence

The narrative is built with 9 shots chained via timeline markers in the `06_CAMERAS` group. Each marker switches the active camera at its frame: orbital opening around the sign (f1), crowd (f301), descent to the ring (f601), introductions of Baki (f961) and Itadori (f1150), three fight angles — wide (f1339), low (f1479) and close (f1620) — and, finally, victory and trophy (f1920). The result is a continuous sequence of about 72 s.

4.4 Crowd

The many crowd instances in the stands play cyclic Mixamo actions in NLA (Sitting Clap/Yell/Angry, Standing Cheering/Clap/Fist Pump, among others). In moments of tension a seated-to-standing transition layer is applied, combining a base pose (`BaseIdle`) with a rise overlay (`RiseSitOverlay`).

4.5 Synchronized audio in the VSE

The sound is assembled in the Video Sequence Editor: a boxing bell at the start and end, a layered crowd bed with crossfades, chants and drums, and a crowd roar at the KO. The `render.use_sequencer` property is deliberately kept OFF: with audio-only strips enabled, the image renders would come out black. The audio is mixed anyway when rendering to video or via a mixdown.

5 Stage 3: Realism and interaction

This stage brings together the materials and textures, the lighting, the cameras and the keyboard interaction layer, along with the consoles' Python scripts and the node graphs that support all of it.

5.1 Materials and textures

The materials follow the **PBR** (*Physically Based Rendering*) model studied in the course: each surface's response to light is described by a BRDF based on microfacet theory, with energy conservation, in which physical parameters such as *base color*, *metallic* and *roughness* replace the old empirical models (Phong/Blinn). In Blender this model is encapsulated by the *Principled BSDF* node. Textures are projected onto the meshes via UV coordinates and the **normal maps** perturb the surface normal per pixel, simulating high-frequency relief without adding geometry — essential in a scene that already approaches 17 million polygons.

The scene gathers 93 materials with this common PBR workflow: they start from the *Principled BSDF* with a color texture in *Base Color* and, where appropriate, a *Normal Map* for relief. Three node-based materials stand out:

- **M_Ring_Canvas** (ring canvas): a **100% procedural** material, with no image textures, built with noise nodes at several scales for the color, the roughness and the micro-relief of the fabric (the full graph is reproduced in the Appendix, Figure 7).
- **M_Ring_UIB_Decal** (UIB logo on the canvas): the logo image feeds both *Base Color* and, through its alpha channel, the BSDF's *Alpha* input, so it appears cut out with no background; a *Mapping* node repositions it without touching the geometry.
- **M_Screen_FightCard** (stadium screen): an **emissive** material that displays the `screen_fight_banner.png` banner and emits its own light, without relying on spotlights.

The character materials (*Body*, *Clothing*, *Hair*, *Eyes* on Baki; set `Male_*_Cos_*` on Itadori) follow the same PBR pattern. They are all packed with *Pack All* into the `.blend`, so the scene is portable.

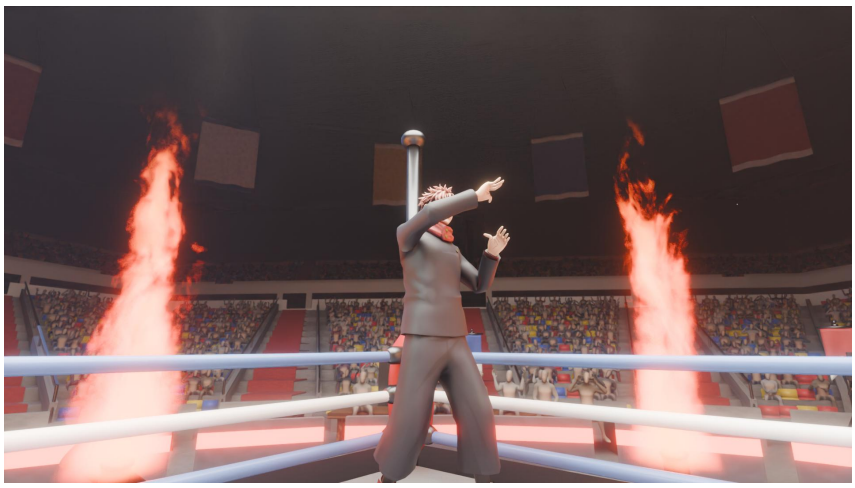

Figure 5: The Yuji Itadori character with his outfit textured via the PBR workflow (*Base Color* and *Normal Map*) and the `Male_*_Cos_*` texture set.

5.2 Lighting

The cinematic's lighting is deliberately *cinematic*: it uses a concert-style key with saturated colors that plays off the rhythm of the fight and changes shot by shot. All the lights are concentrated

in the outliner's `04_LIGHTING` group and are animated to synchronize each lighting state with the active camera. The set is organized into families with distinct functions:

- **Colored concert spotlights** that sweep the stands: `Spot_Concert_` (Blue, Cyan, Green, Magenta, Orange, Pink) and `Spot_Lower_` (Aqua, Hotpink, Lime, Purple, Red, Yellow). They provide the show atmosphere and bring the crowd to life during the opening and introduction shots.
- **Ring spotlights** `FIGHT_RingSpot_`, in blue and white variants, that concentrate attention on the canvas in the fight shots.
- **Key light** `KeyLight_Top`, which ensures a consistent key on the characters, and a **trophy light** `TROPHY_KeyLight`, reserved for the final victory shot.
- **Complementary groups** also controlled from the console: the glow of the big screen, the general crowd *wash*, the *rim* that outlines the ring's contour and the *reveal* that uncovers it in the transition to the fight.

The lighting is not static: it evolves across the nine shots, reinforcing the narrative and guiding the viewer's eye.

From a theoretical standpoint, Cycles is a **path tracing** engine: it solves the rendering equation by Monte Carlo integration, tracing rays from the camera that bounce through the scene and sample the light sources. This provides **global illumination** — indirect light, soft area shadows and color bleeding between surfaces — in a physically plausible way, unlike EEVEE's rasterization, which we use only as a preview. The price is sampling noise, which decreases with the number of samples per pixel; hence the optimization decisions taken: per-shot sample *keyframes* (more samples only where the shot requires it) and *caustics* disabled (the scene has no glass), cutting render time with no visible loss.

5.3 Cameras

The nine cameras of the `06_CAMERAS` group resolve the cinematic as a continuous long take. Each has its framing, focal length and, in some cases, position animation; the active camera is switched by timeline markers, so playback automatically runs through the entire narrative with no manual intervention.

5.4 Keyboard interaction (no Game Engine)

Unlike a video game, this project delivers a cinematic, but it retains an interactive layer. The main design constraint is that Blender 4.x **does not include a Game Engine**: the sensors and actuators of the old BGE no longer exist. Therefore, interaction is not built with logic blocks, but with **custom Python modal operators** (light and camera consoles) and with the program's native playback control.

The console is implemented as a custom addon that registers the `uib.light_console` operator and adds a panel in the "UIB" tab of the Viewport sidebar. It is activated with the **Ctrl+Shift+L** shortcut. While the modal mode is active, the keyboard controls the scene's eight lighting groups live:

- Keys `[1]`–`[8]`: toggle each of the 8 light groups on or off.
- Key `[0]`: turns all groups on at once.
- Key `[Esc]`: closes the console and ends the modal operator.

A key detail is that each toggle modifies **both** `hide_viewport` and `hide_render` of every light in the group: what is turned off in the viewport is also turned off in the render, so what the user sees matches what would be rendered.

Alongside it, a **Camera Console** (`uib.camera_console`, shortcut `Ctrl+Shift+C`) allows jumping between the nine shots with keys `[1]`–`[9]` or the arrow keys, setting each shot's camera and frame without scrubbing the timeline. Finally, the **space bar** plays or pauses the entire cinematic.

5.5 Python scripts and nodes

The programmed functionality consists of the two modal operators described above, `light_console.py` and `camera_console.py` (in `blender-project/scripts/`). The node logic (UIB logo decal and emissive screen) is described in the materials and textures section.

6 Stage 4: Advanced realism

This stage covers the advanced realism techniques implemented in the scene, all concentrated in the outliner's `05_FX` group.

6.1 Pyrotechnics and emissive fire

Flame throwers (`PYRO_Burst_N`, `E`, `S` and `W`) have been placed at the four corners of the ring, each with its modeled housing and a set of emissive flames organized in layers: `Flame_Outer`, `Flame_Inner`, `Flame_Core` and `Flame_Tongue`. These flames emit their own light through materials with an emission node, which adds a localized warm light consistent with the cinematic lighting.

6.2 Volumetric smoke

The `FX_Smoke_Atmosphere` set adds a volumetric atmosphere to the scene. It includes a ground haze (`FX_GroundHaze`) that gives depth to the ambiance, corner smoke in red and blue tones that reinforces the chromatic division of the confrontation, and the smoke associated with the pyrotechnics themselves. The use of volumetrics lets the light from the spotlights and the flame throwers scatter realistically through the arena's air.

6.3 Particles, physics, gravity and collisions

Two particle systems governed by Newtonian physics (gravity included) have been implemented:

- **Dust motes** (`DustMotes`): 260 particles suspended in the air that catch the light and add a sense of volume to the space.
- **Confetti** (`Confetti`): two emitters of 2,400 particles each, placed in the cannons, that fire in the victory shot. A collision object (`Confetti_GroundCollision`) is placed on the floor so the particles accumulate plausibly instead of passing through the geometry.

In total, particles, physics, collision, gravity, volumetrics and emissive fire have been implemented, covering the advanced techniques required by the assignment.

Figure 6: Victory shot with the confetti cannons firing, the corner pyrotechnics and the volumetric atmosphere.

7 Stage 5: User evaluation

To assess the ease of use of the project's interactive part, a usability evaluation was carried out using a ten-item questionnaire adapted from the SUS (*System Usability Scale*). The evaluation focused on the custom components the user interacts with: the two keyboard consoles (light and camera) described in Stage 3.

7.1 What is evaluated

The questionnaire measures the **ease of use of the project's interactive part**, which the user interacts with when opening the scene in Blender:

- The **Light Console** (`Ctrl+Shift+L`): keys `[1]`–`[8]` to toggle light groups, `[0]` all, `[Esc]` to exit.
- The **Camera Console** (`Ctrl+Shift+C`): jump between the 9 shots with `[1]`–`[9]` or the arrow keys.

Throughout the questionnaire, "the system" refers to this interactive part.

7.2 The questionnaire

The instrument that each participant filled in is reproduced below.

Participant details

Identifier: .....

Prior experience with Blender: ☐ None ☐ Low ☐ Medium ☐ High

Date: .....

Instructions

For each statement, mark your level of agreement on a scale of **1 to 5**, where **1 = Strongly disagree** and **5 = Strongly agree**. Answer spontaneously; there are no right or wrong answers.

The 10 items

#	Statement	1	2	3	4	5
1	I think I would use this system frequently.	☐	☐	☐	☐	☐
2	I find the system easy to understand.	☐	☐	☐	☐	☐
3	I think the system is easy to use.	☐	☐	☐	☐	☐
4	I think I could use the system without the help of a technician.	☐	☐	☐	☐	☐
5	The system's functions are well integrated.	☐	☐	☐	☐	☐
6	I think the system is consistent.	☐	☐	☐	☐	☐
7	I imagine that most people would learn to use the system very quickly.	☐	☐	☐	☐	☐
8	I find the system comfortable to use.	☐	☐	☐	☐	☐
9	I feel very confident using the system.	☐	☐	☐	☐	☐
10	I was able to start using the system without having to learn a lot beforehand.	☐	☐	☐	☐	☐

Free comments

.....

7.3 Methodology

The above questionnaire was administered to eight participants. Each participant opened the scene in Blender, performed the basic tasks (activating the light console, toggling several groups and jumping between shots with the camera console) and then filled in the ten-item questionnaire, all worded positively and answered on a scale of 1 to 5.

7.4 Score calculation

Since all items are worded positively (the greater the agreement, the better the usability), each participant's score is the **sum of their ten answers** (each from 1 to 5), in the range 10–50: the higher, the better.

7.5 Results

Table 1 shows the scores of the eight participants.

Participant	P1	P2	P3	P4	P5	P6	P7	P8	Mean
Score	45	40	46	41	43	38	45	42	42.5

Table 1: Usability scores per participant (sum, out of 50).

Mean = 42.5 out of 50 $\quad (n = 8)$.

7.6 Interpretation

The mean obtained is 42.5 out of 50, which, normalized to the usual SUS scale (0–100), corresponds to a score of **85**. Taking the literature values as a reference — the mean of systems evaluated with SUS sits around 68, and above 80 is considered an excellent result ("A" range) — the interactive part clearly stands above the acceptability threshold.

Beyond the mean, the **spread is low**: the scores range from 38 to 46 (an 8-point range), so even the most critical participant scored well above the scale's neutral midpoint (30 out of 50). There is therefore no "lost" user: the result is consistent across profiles with different prior experience in Blender, which we attribute to the consoles using immediate conventions (number keys, `Esc` to exit) and showing an on-screen HUD with the state of each group, so the system is self-describing. Participants particularly valued the consistency between what they see and what is rendered: turning off a light in the viewport also turns it off in the render.

As limitations, the sample is small ($n = 8$) and of a similar profile (students), so the results are indicative but do not generalize; moreover, the adapted questionnaire uses all ten items positively, which simplifies the calculation but gives up the acquiescence control of the original SUS (items alternating between positive and negative).

The evaluation also served its **formative** purpose: following one comment — that the second shot (the crowd one) lacked colored lights — more concert spotlights were added to that shot, an improvement incorporated into the scene before submission.

8 User manual

This section describes, step by step, how to open the project, play the cinematic, interact with the UIB consoles (light and camera) and produce a final video. The manual assumes a machine with Blender 4.5 installed.

8.1 Prerequisites

- Blender 4.5 with the Cycles engine available. The viewport preview can be worked in EEVEE for greater fluidity.
- No additional asset installation is required. All `.blend` files are packed with *Pack All*: the textures (93 materials) and the audio travel inside the file, so the scene opens with no broken paths on any machine.
- Recommended hardware: the scene is heavy (on the order of 18,800 objects and about 16.7 million crowd polygons). It is best to work the viewport in *Material Preview* mode, never in *Rendered*.

8.2 Opening and playback

1. Open `blender-project/final_scene.blend`.
2. Press the **space bar** to start or pause playback. The cinematic spans frames 1 to 2176 at 30 fps (about 72 s) and chains 9 shots.
3. To navigate between shots, use the timeline's **marker keys**: each shot jumps to its camera via a timeline marker, from `Cam_01_Sign` (opening) to `Cam_06_Trophy` (victory and trophy).

8.3 Keyboard controls: UIB consoles

The interactive part is controlled with two custom Python modal operators, each with its button in the **UIB** panel of the viewport sidebar and its shortcut. The **Light Console** (`Ctrl+Shift+L`) *modifies the scene* (changes visibility in the viewport and render at once); the **Camera Console** (`Ctrl+Shift+C`) jumps between shots. Both use `[1]`–`[9]` to select and `[Esc]` to close; the light one also uses `[0]` to turn everything on. The Light Console groups are summarized in Table 2.

Table 2: Controls of the UIB Light Console.

Key	Action
1	Toggle the concert spotlights (Concert spots)
2	Toggle the pyrotechnics (Pyro fire)
3	Toggle the ring *rim* (Ring rim)
4	Toggle the stage spotlights (Stage spots)
5	Toggle the key light (Key light)
6	Toggle the screen glow (Screen glow)
7	Toggle the crowd *wash* (Crowd wash)
8	Toggle the ring *reveal* (Ring reveal)
0	Turn on all groups
Esc	Close the console

8.4 Camera switching and triggering animations

Camera switching is automatic during playback: each timeline marker switches to its shot's camera. In addition, the **Camera Console** (`Ctrl+Shift+C`) allows manually jumping to any

of the nine shots with keys `[1]`–`[9]` or the arrow keys, setting its camera and frame without scrubbing the timeline. The animations of characters, crowd, lights and effects are all on the timeline, so simply playing (space bar) triggers them.

8.5 Rendering to video

To export the cinematic as a video, set the output to a video format and launch the animation render (Cycles, 1920x1080, with per-shot sample keyframes). The audio synchronized in the Video Sequence Editor is mixed automatically when rendering to video or via a *mixdown*. Important: the `render.use_sequencer` property is deliberately OFF; if it were enabled with only audio strips in the sequencer, the image renders would come out black.

9 Advantages and drawbacks

Advantages. The eight-group outliner organization and the nine-camera marker-driven long take make an enormous file manageable and yield a continuous cinematic, playable inside Blender. The interaction is custom and consistent (the console changes `hide_viewport` and `hide_render` at once, without a Game Engine), and *Pack All* packing makes the scene portable.

Drawbacks and difficulties. Blender 4.x does not include a Game Engine (interaction was solved with Python operators); Mixamo retargeting caused *clipping* in Itadori's arms (fixed with a rest-pose retarget) and the armatures hang from scaled *empties*; the slotted *Actions* of 4.5 complicate adjusting the lights by code; and the crowd's weight ($\sim$16.7 M polygons) forced the use of LOD, thinned only in the viewport, and controlling render cost (samples per shot and *caustics* disabled).

10 Critical assessment of the work

The project consolidated Mixamo retargeting, Blender 4.5's slotted *Actions*, levels of detail, volumetrics and physics-driven particles, and programming the interaction in Python. The most decisive factors for keeping control were the group organization, the camera markers and *Pack All* packing. Overall, the course's theoretical concepts — transformation hierarchies, PBR, cinematic lighting, sampling in *path tracing* and particles — were applied directly, conditioned by real performance constraints.

11 Conclusions

The project delivers a boxing cinematic of about 72 s (2176 frames, 30 fps, 1920×1080, Cycles) that integrates all the requirements of the assignment: modeling of the arena, two animated characters, a large crowd, cinematic lighting, materials and textures, advanced techniques (particles, physics and volumetrics) and a custom interactive layer, on a scene of ∼18,800 objects. All the required stages are covered and the usability evaluation of the interactive part was positive (mean 42.5 out of 50).

12 Appendices

12.1 Python code

The complete code of the two interaction modal operators (`light_console.py` and `camera_console.py`, ∼180 lines each) is in `blender-project/scripts/`. The key fragments are reproduced here.

12.1.1 Light Console: groups and toggling

Each light group is defined by name prefixes, and the toggle changes visibility in the viewport and render at once:

```
# (label, prefixes) -- a LIGHT belongs to the group when its
# name starts with one of the listed prefixes.
LIGHT_GROUPS = [
    ("Concert spots", ("Spot_Concert_", "Spot_Lower_")),
    ("Pyro fire",     ("PYRO_",)),
    ("Ring rim",      ("CINE_Ring_Rim_",)),
    ("Stage spots",   ("StageSpot_",)),
    ("Key light",     ("KeyLight_",)),
    ("Screen glow",   ("CINE_Cam03_Screen_",)),
    ("Crowd wash",    ("CINE_Back_Crowd_",)),
    ("Ring reveal",   ("CINE_Cam03_Ring_Reveal_",)),
]

def _group_lights(prefixes):
    return [o for o in bpy.context.scene.objects
            if o.type == 'LIGHT' and o.name.startswith(prefixes)]

def _set_group(prefixes, on):
    for o in _group_lights(prefixes):
        o.hide_viewport = not on
        o.hide_render = not on   # viewport and render always consistent
```

12.1.2 Light Console: modal loop

The modal operator captures the keyboard while active: keys `1`–`8` toggle groups, `0` turns everything on and `Esc` exits; the rest of the events pass through to Blender (`PASS_THROUGH`):

```
class UIB_OT_light_console(bpy.types.Operator):
    bl_idname = "uib.light_console"
    bl_label = "UIB Light Console"

    def modal(self, context, event):
        if event.type in {'ESC', 'RIGHTMOUSE'} and event.value == 'PRESS':
            self._finish(context)
            return {'CANCELLED'}
        if event.value == 'PRESS':
            if event.type in _KEYS:            # keys [1]-[8]
                idx = _KEYS[event.type] - 1
                if idx < len(LIGHT_GROUPS):
                    label, on = _toggle_group(idx)
                return {'RUNNING_MODAL'}
            if event.type in {'ZERO', 'NUMPAD_0'}:
                for _, prefixes in LIGHT_GROUPS:
                    _set_group(prefixes, True)
                return {'RUNNING_MODAL'}
```

```
        return {'PASS_THROUGH'}
```

12.1.3 Camera Console: reading shots and jumping

The shots are read live from the timeline's camera markers (sorted by frame), so the console always reflects the file's actual cameras; jumping to a shot sets its frame, its camera and the camera view:

```
def _shots():
    # Markers with a camera, sorted by frame -> one shot each.
    mk = [m for m in bpy.context.scene.timeline_markers if m.camera]
    mk.sort(key=lambda m: m.frame)
    return mk

def _go(context, marker):
    sc = context.scene
    sc.frame_set(marker.frame)
    sc.camera = marker.camera
    area = context.area
    if area and area.type == 'VIEW_3D':
        area.spaces.active.region_3d.view_perspective = 'CAMERA'
```

12.1.4 Registering the keyboard shortcut

Each console is registered as an addon with its shortcut (`Ctrl+Shift+L` / `Ctrl+Shift+C`) and its panel in the "UIB" tab of the sidebar:

```
def register():
    bpy.utils.register_class(UIB_OT_light_console)
    bpy.utils.register_class(UIB_PT_light_console)
    kc = bpy.context.window_manager.keyconfigs.addon
    if kc:
        km = kc.keymaps.new(name='3D View', space_type='VIEW_3D')
        kmi = km.keymap_items.new("uib.light_console", 'L', 'PRESS',
                                  ctrl=True, shift=True)
        _addon_keymaps.append((km, kmi))
```

Both consoles also draw a HUD over the viewport (`blf` module) with the state of each group or shot, registered as a *draw handler* of `SpaceView3D` while the operator is active.

12.2 Node setups used

The scene's notable node-based materials can be inspected by opening each material in the *Shading* editor of `final_scene.blend` (select the object — the ring canvas or the stadium screen — and its material in the editor header).

12.2.1 M_Ring_Canvas (ring canvas, procedural material)

A **100% procedural** material: it uses no image texture, so it depends on neither UV coordinates nor resolution. Starting from the *Object* coordinates (*Texture Coordinate* node), four *Noise* nodes (fBM) at different scales generate the variation of the fabric: the low-frequency ones, through a *Color Ramp* and mix and multiply nodes, produce the stains and wear of the *Base Color*; others, tuned with *Map Range* nodes, control the *Roughness*; and the finest noise (scale ∼130, the weave of the fabric) feeds a *Bump* node connected to the *Normal* input of the *Principled BSDF*, creating the canvas's micro-relief with no additional geometry (Figure 7).

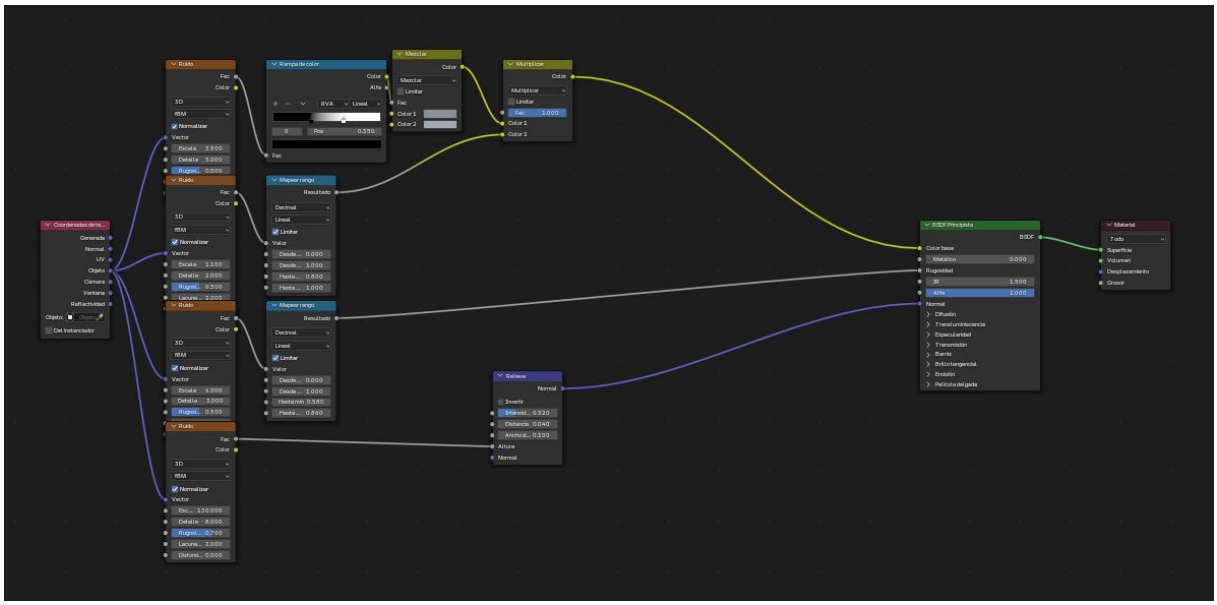

Figure 7: Node graph of the procedural material `M_Ring_Canvas` of the ring canvas: fBM noises at several scales for color (via Color Ramp and mix), roughness (via *Map Range*) and micro-relief (via *Bump* → *Normal*).

12.2.2 M_Ring_UIB_Decal (UIB logo on the canvas)

Node chain: *Texture Coordinate* (UV) → *Mapping* (repositions and scales the logo without touching the geometry) → *Image Texture* (`uib_logo`) → *Principled BSDF* → *Material Output*. The texture's *Color* output feeds *Base Color* and its *Alpha* channel enters the BSDF's *Alpha* input, so the logo appears cut out on the canvas with no background.

12.2.3 M_Screen_FightCard (emissive stadium screen)

Node chain: *Image Texture* (`screen_fight_banner.png`) → *Emission* (the texture as emission color, with its intensity in *Strength*) → *Material Output*. Being an emissive material, the screen emits its own light onto the scene — it is a textured area light that Cycles samples like any other — without relying on auxiliary spotlights.

12.3 Bibliography, resources and licenses

Software

- **Blender 4.5** (modeling, animation, Cycles rendering, VSE and the `bpy` Python API) — GNU GPL license — `https://www.blender.org`

Animations

- **Adobe Mixamo** — animation clips for the boxers (*Presentation*, *Rockin' Shackle*, *Fist Fight*, *Victory*, *Sad Idle*) and the crowd (*Sitting Clap/Yell/Angry/Disbelief/Idle/Pointing*, *Standing Cheering/Clap/Fist Pump*), retargeted to our own skeletons. Free to use under Adobe's terms of service — `https://www.mixamo.com`

Models and textures

- Character models and textures, obtained from Sketchfab; used exclusively for academic purposes in this project, with no redistribution or commercial use. The characters belong to their respective authors (*Baki the Grappler*, K. Itagaki; *Jujutsu Kaisen*, G. Akutami):

  - **Baki Hanma** — `https://sketchfab.com/3d-models/baki-hanma-v10-bb74f3a66c9b4b60a300d2b9a61a2737`
  - **Yuji Itadori** — `https://sketchfab.com/3d-models/yuji-itadori-free-fire-skin-6478cf5726354f3a9393b6ce1bfe6e9b`
- Set and *branding* textures (canvas, screen, signage): our own work, including the `screen_fight_banner.png` banner; the UIB logo is property of the Universitat de les Illes Balears and is used for academic purposes.

Audio

The sound effects come from the following sources:

- **Mixkit** — crowd beds and reactions, applause and chants, under the *Mixkit Sound Effects Free License* (free to use, no mandatory attribution) — `https://mixkit.co/license/#sfxFree`
- **Red Library** (via the Internet Archive, collection `Red_Library_Crowds_Sports`) — boxing bell, public domain CC0 1.0 — `https://creativecommons.org/publicdomain/zero/1.0/`

Project file	Original sound	Source
`amb_crowd_bed.mp3`	Ambient sports crowd sound (id 2097)	Mixkit
`crowd_stadium_alt.mp3`	Crowd at the stadium (id 2111)	Mixkit
`crowd_chants_drums.mp3`	Stadium chaotic loud applause, drums and chants (id 363)	Mixkit
`crowd_applause_loop.mp3`	Applause ambience loop (id 513)	Mixkit
`crowd_applause_cheer.mp3`	Rhythmic applause and cheering (id 504)	Mixkit
`crowd_applause_moderate.mp3`	Auditorium moderate applause and cheering (id 502)	Mixkit
`crowd_clap_rhythmic.mp3`	Rhythmic audience clapping loop (id 522)	Mixkit
`crowd_cheer_whistle.mp3`	Cheering crowd loud whistle (id 610)	Mixkit
`crowd_cheer_short.mp3`	Male crowd cheering short (id 459)	Mixkit
`bell_boxing.wav`	R07-13 Crowd and Boxing Bell (CC0 1.0)	Red Library
`crowd_goal_roar.m4a`	Crowd roar at the KO, contributed to the project	—

Bibliography

- Course notes and materials for 21714 Computer Graphics, UIB, 2025–2026 academic year.
- Blender Foundation. *Blender 4.5 Manual* (Cycles, shading nodes, NLA, particle systems, Python API) — `https://docs.blender.org`
- Brooke, J. (1996). "SUS: A quick and dirty usability scale". In *Usability Evaluation in Industry*. Taylor & Francis. (Basis of the Stage 5 questionnaire.)
- Pharr, M., Jakob, W., Humphreys, G. *Physically Based Rendering: From Theory to Implementation*. (Reference for the PBR model and path tracing.)