ChatGPT

Taitaja Pelituotanto – Yearly Task Analysis and 2026 Prediction

Year-by-Year Final Task Breakdown

2019 (Joensuu) – **3 modules / tasks** (Pre-production, Production, Presentation). Total time ~24h over 3 days.
- **Task 1 (Moduuli 1: Esituotanto)** – *Game concept design*. ~8h allotted. Teams produced a game **design document** and concept art for a **tower-defense strategy game with a twist** (easy to play, hard to master). No specific engine mandated (Unity/Unreal mentioned as options).
- **Task 2 (Moduuli 2: Tuotanto)** – *Game prototype development*. ~11h total (Day 2 + part of Day 3). Teams built a **playable prototype** demonstrating key mechanics (e.g. towers, enemies, waves) in line with the design. Both 2D and 3D graphics were allowed. Deliverable included the **game project files and a runnable build**.
- **Task 3 (Moduuli 3: Esittely)** – *Presentation & demo*. ~5h prep on Day 3. Teams created a **PowerPoint/ Keynote presentation** (understandable standalone) covering the game concept, mechanics, etc., and presented it to judges. Presentation content had to align with the prototype and include the original concept art and design choices.

2021 (online/Helsinki) – **3 modules** (likely A, B, C). Total time ~8h (one-day final, due to remote format).
- **Task 1 (Design & Planning)** – Teams drafted a game concept and basic plan. A *secret theme* was introduced in the assignment (to prevent pre-built projects), encouraging creative interpretation.
- **Task 2 (Prototype Development)** – Teams developed a small **game prototype** around the given theme within the same day. Likely focus on implementing core gameplay mechanics and minimal content.
- **Task 3 (Presentation)** – Teams prepared a short presentation of the concept and demo. Given the remote nature, emphasis was on clarity of presentation materials. *(Detailed 2021 task docs were encoded and difficult to read, but structure appears consistent with other years.)*

2022 (Pori) – **3 modules** (named A, B, C). Total ~24h over 3 days. **Introduced new WorldSkills-aligned structure and "secret theme."**
- **Module A – "Game Project"** (9 points): *Project setup & organization*. Evaluated end of Day 1. Teams had to organize their Unity/Unreal project using **good versioning and naming practices** (logical folder structure, assets named consistently). Only ~1 day was devoted to initial planning in this year.
- **Module B – "Game Build"** (58 points): *Gameplay implementation*. Evaluated end of Day 2. This was the core coding task – teams implemented the game on **Windows PC** following the secret theme. **Mandatory mechanics** included intuitive controls, working core mechanics, a fun gameplay loop, a smooth camera, and in-game instructions for the player. The game had to run without frame drops on the provided test PCs. **UI and graphics** were evaluated for clarity and appeal – an attractive art style fitting the theme, clear visuals, and a readable user interface were expected. Teams needed to include **audio** (music and sound effects) that fit the mood/theme. *Performance and polish were explicitly rewarded:* e.g. no significant lag, coherent color palette, etc. Accessibility criteria appeared for the first time – e.g. colorblind-friendly design and readable font sizes (worth ~3 points).
- **Module C – "Presentation & Game Design"** (33 points): *Presentation and design rationale*. Evaluated Day 3 after final presentations. Teams submitted a **PowerPoint presentation in English** explaining how the game works and how it meets the theme. Key points judged: visual quality of slides, use of the

game's color scheme in the presentation, a catchy game name/tagline (with an entrepreneurial mindset), manageable amount of text per slide, and absence of typos. The **oral presentation** was scored on clarity of speech and how well the team explained the game's core mechanics, innovative features, the **use of the secret theme**, the intended player experience (MDA: mechanics → dynamics → aesthetics), and target audience. Notably, **game design documentation** was folded into this module – teams had to articulate their design decisions here, rather than in a separate Day 1 doc.

(2022 Notes:) The **secret theme** was revealed at the start of the competition and applied to all modules. Internet use was allowed for research (asset downloads, documentation) but any external communication or asking for help was forbidden. Use of premade assets (Unity Asset Store, Unreal Marketplace packages, free graphics/audio) was permitted with proper licensing, but teams were expected to understand and integrate them appropriately. Copy-pasting code without understanding was discouraged – clear code commenting was noted as evidence of understanding. Teams worked in pairs and had to submit their work in a single folder with subfolders for each module (to streamline judging). All final deliverables (project files, source code, assets, executables, and presentation) were included in that folder. The competition task had to be **performed and presented in English**, even though instructions were given in Finnish.

2023 (finals) – **3 modules** (presumed). *Details are sparse* (only an evaluation sheet was provided). The structure likely mirrored 2022 with minor tweaks: ~100 points total, multi-day. We can infer:
- **Module A:** Project setup / initial concept (possibly ~10–15% weight).
- **Module B:** Game prototype build (~55–60% weight). Criteria probably similar to 2022 – judging gameplay, technical functionality, visuals, UI, audio. Accessibility and theme integration likely remained important.
- **Module C:** Presentation & game design justification (~25–30% weight). Focus on how well the team presented their game and explained design decisions in English. *(Without the task brief, the exact theme or genre for 2023 is unknown, but it followed the pattern of giving a unique theme to base the game on.)*

2024 (Jyväskylä) – **3 modules**. Total ~24h over 3 days.
- **Module 1: Pre-production (20 p)** – *Game Design Documentation*. Day 1 was devoted to planning. Teams received a **client brief**: *"Kestävän kehityksen videopelin prototyyppi"* – a **sustainable development** themed game prototype. They had to outline the game concept fulfilling this theme (genre could be simulation, puzzle, or strategy per the brief) and produce a **design document** for the idea. The design plan (and any concept art or document) was submitted end of Day 1 for scoring. Judges expected the plan to reflect the client's requirements and a creative interpretation of the sustainability theme. Notably, teams were encouraged to apply the **MDA framework** (Mechanics-Dynamics-Aesthetics) in their design process – showing a more advanced game design approach was valued.
- **Module 2: Production (60 p)** – *Game Development*. Spanning Day 1 (afternoon) + Day 2. Teams began prototyping as soon as the design doc was done, and the bulk of coding/art/audio happened in this module. All core gameplay had to be implemented by end of Day 2. The expected content was similar to prior years: a functional game level or loop that addresses the theme (**sustainability challenges and solutions** in this case), with polished mechanics, camera, UI, graphics, and sound. Judges evaluated playability (controls, mechanics), technical performance, visuals, and audio as usual. Additionally, **project structure and code quality** were likely checked (though 2024's marking form suggests those might have been assessed qualitatively).
- **Module 3: Post-production (20 p)** – *Presentation & Delivery*. Day 3 focused on preparing the final presentation and delivering it. Teams presented the prototype as if to the "client," explaining how the game addresses sustainable development. The presentation module emphasized professional delivery and coherence with the prior modules. **Crucially, 2024 stressed consistency:** what was designed on Day 1, built on Day 2, and presented on Day 3 had to align. Teams were explicitly warned that all three modules should connect – the final product should reflect the initial plan. (In other words, you shouldn't

abandon your Day 1 concept; any changes had to be justified.) As in 2022, slide quality, theme explanation, and overall communication were scored. The presentation was done in English and had to highlight the game's theme compliance, mechanics, innovation, etc., within a polished slide deck.

(By 2024, the client-based prompt and emphasis on real-world themes indicated that the task was moving closer to WorldSkills standards. The points distribution was balanced (20/60/20), giving equal weight to planning and presentation. This suggests that both design and *presentation skills were as crucial as coding skills in the judging.)*

2025 (Espoo) – **3 modules**. Total time ~18 hours over ~2.5 days (slightly compressed schedule).
- **Module 1** – *Concept Plan*. Only ~2.5 hours were given on Day 1 for concept design – much shorter than previous years. Teams received a specific client brief: *"A Culinary Deck Builder"* – the client wanted a **single-player card game about cooking**. In that brief, the core concept and genre were defined (a cooking-themed deck-building game), so teams focused on designing the gameplay details and style. By 11:00 on Day 1 they submitted a short **design document** (game concept and plan). Despite the brief time, teams needed to outline the card game mechanics (e.g. cards representing ingredients/recipes, how cards are drawn/played) and how it meets the client's expectations.
- **Module 2** – *Development*. Spanning the remainder of Day 1, all of Day 2, and ending by Day 2 afternoon (~13.5 hours total). Teams implemented the **playable card-game prototype**. Mandatory features included the core deck-building mechanics (shuffling/drawing cards, playing cards to cook or score), a working gameplay loop (possibly multiple rounds or levels of increasing difficulty), and basic UI to display the cards, scores or outcomes. Given the genre, this module likely required **extensive UI programming** (for card interfaces) and game logic (turn or phase-based mechanics) rather than character movement. As always, the build had to be stable and include instructions (controls/rules) for the judges. Visuals and audio were still evaluated – e.g. card artwork or placeholders that fit the cooking theme, and sound effects or music to enhance the mood. Teams that prepared reusable card-game scripts or knew Unity 2D UI well would have an advantage here.
- **Module 3** – *Presentation*. Day 3 morning (~1.5–2 hours prep), followed by live presentations to the judging panel. Teams compiled a **presentation (slides)** explaining how the "culinary deck builder" works, its unique selling points, and how it met the client's requirements. Likely points of emphasis: clarity in explaining rules of the card game, how the theme (cooking) is integrated (e.g. did the game teach recipes, or just use cooking as story?), and the completeness of the prototype. Common criteria such as slide readability, concise text, and speaking clarity were applied (similar to previous years). Because the genre was predefined, judges probably paid attention to **game design innovation** – e.g. what twist did the team add to make their cooking card game stand out. Any mismatch between what was promised in the design doc and what was delivered in the prototype would have been noted (consistency across modules remained important).

(2025 Notes:) The continued trend of a **client-based task prompt** was evident. The narrowed theme (specific genre + theme combo) tested the competitors' ability to execute a well-known game type (deck-builder) with a creative theme (cooking) under time pressure. The schedule shows an even **greater emphasis on development time** and less on upfront planning – possibly assuming competitors are now faster at drafting concepts. Module1 was just long enough to outline a design, then teams jumped into coding. The expectation was that by now, teams come prepared with **template solutions** for common systems (for example, having practiced a simple card game or inventory system beforehand). The judging likely still followed the balanced scoring (roughly 15–20% concept, ~60% prototype, ~20% presentation), though 2025's exact point split isn't explicitly given. Quality of execution (functionality and polish) was heavily rewarded, as even a simple concept can earn high marks if well executed.

Semifinals (2019–2025) – Preliminary rounds followed a similar structure but compressed into a single day (8 hours total). Typically, the semifinal task is a **mini version of the final**: competitors (often solo in

semis) must design a quick game concept and develop a small prototype by day's end, then give a short presentation. For example, in 2019 the semifinal was structured in 3 modules (Pre-production, Production, Presentation) all within one 8h day. Semifinal game prompts tend to be simpler "starter game" scenarios (e.g. basic platformer, endless runner, simple shooter or puzzle) to test fundamental skills. The **mandatory mechanics** in semifinals are usually very basic: e.g. implement player movement and jumping, at least one enemy or obstacle, a scoring or timer system, and a start/end screen. Semifinal evaluation focuses on **functionality over polish** – does the game loop work? Are there win/loss conditions? Did the competitor include instructions and basic UI (score/health)? Visuals and audio carry less weight in semis but can be tiebreakers. Common semifinal pitfalls are failing to complete a playable level or having game-breaking bugs due to the tight timeline. (Notably, semifinal tasks often reuse themes or mechanics from *older finals* as practice – a hint that mastering fundamentals featured in past finals is good preparation for semis.)

Cross-Year Patterns and Trends

- **Consistent Task Structure:** Every year's **final** has three main tasks/modules: **(1) Plan & Design, (2) Develop Core Game, (3) Present & Polish**. This structure is now standard, aligning with the idea of a game development cycle (pre-production → production → post-production). Semifinals mirror this on a smaller scale.
- **Theme-Based "Client" Briefs:** Each competition introduces a unique theme or genre twist that all teams must incorporate. **All years require designing around a given theme.** For example, 2019's theme was a *tower-defense game with a twist*, 2024 focused on a *sustainability-themed game*, and 2025 specified a *cooking-themed card game*. Some years the theme is revealed on the spot as a **"secret theme"** to ensure no one pre-builds their game. What rotates is **the genre and subject matter** – one year it might be strategy, next year a puzzle or card game. However, the task always demands a creative interpretation of the theme and evidence that the theme influenced the design (judges typically allocate ~10% of points just to how well the theme was used/executed).
- **Core Gameplay Mechanics:** Despite different themes, the *fundamental mechanics required* show strong overlap year to year. Every final project must have:
- **Player interaction/control** – e.g. character movement or some input mechanism. Judges always check that controls feel intuitive and responsive. Even in a card game (2025), this translates to a smooth UI for selecting/playing cards.
- **A functional gameplay loop** – The prototype should be "playable" in the sense of having repeatable actions or levels and a clear objective or win condition. Whether it's surviving waves in a tower defense, completing puzzles, or playing through cards, the game needs a loop that's *fun and engaging*. An endless mode or a simple win/lose condition is expected.
- **At least one enemy or challenge element** – e.g. AI enemies in a shooter or tower defense, hazards in a platformer, or competitive mechanic in a card game. The game can't be just a sandbox; there should be something to overcome or manage (this is often implicitly required by the theme/genre).
- **Scorekeeping or progression** – Most years expect some scoring system, health bar, timer, or progress indicator as appropriate to the game. While not always explicitly listed, **UI elements** like score, timer, health, or inventory are indirectly checked under "UI is clear and informative" criteria. In short, the player should be able to track their status.
- **User Interface & Instructions** – Every game must include basic UI and **player instructions**. It is repeatedly mandated that the build contain all info needed for a judge to play: e.g. a controls screen or tutorial prompt. A start menu or pause menu isn't explicitly required, but teams often include a start screen with instructions (easy way to score the "instructions to player" points). UI is judged both on functionality (clarity, all necessary info present) and on aesthetics (cohesive with game art).

- **Audio** – All finals expect the game to have sound. Music and sound effects are consistently part of scoring, usually totaling ~5–10% of points. The **mood fit** and appropriateness of audio are evaluated. Absolute silence in the game will lose easy points – even simple free sounds or music loops are better than none.
- **Basic performance stability** – The game is expected to run **without crashes or severe frame-rate issues** on the provided hardware. Every year's criteria penalize unplayable builds. Using lightweight assets and optimizing any expensive operations (or keeping scope small) is important given the limited time to test performance.
- **Deliverables and Platforms:** The required deliverables have remained constant. Teams must **deliver a Windows PC executable build** of the game plus all project files, documents, and assets. The platform is always a Windows desktop – mobile or web builds are not asked for. This means competitors can count on using Unity or Unreal on PC. There's an implicit expectation to use a **common engine (Unity is most popular)** – multiple docs mention *"Unity/Unreal Engine"* as the context. While any engine is allowed in theory, using something obscure might risk compatibility or lost time. All assets (including those downloaded) must be included and properly organized. Judges consistently emphasize **folder structure and naming** (for ease of review) – e.g. separate folders for code, art, audio; intuitive names. Deliverables also always include the **design document** (at least for modules that require it) and the **presentation slides**. Naming conventions for files and submission folders are given to avoid confusion, and not following them can irritate judges (if not formally penalized, it could affect impression).
- **Evaluation Criteria Emphasis:** The scoring each year totals 100 points, broken into criteria that cover: **project management**, **game functionality**, **game content (art/audio)**, and **presentation/communication**. A pattern observed:
- **Gameplay/Functionality is king** – Roughly half or more of the points are always tied to the game working well and being fun: controls, mechanics, lack of bugs, performance, etc. If the core game doesn't function, teams lose a huge chunk of points. This means a *simple but working game* often beats an ambitious but broken one.
- **Visual polish and UX** – Around 20–30% of points relate to visuals (graphics style, clarity, UI design) and user experience. Teams don't need artist-level custom graphics to score well, but they do need a **cohesive look and clear presentation** in-game. Judges reward games that *"look and feel like a complete product"* within the constraints – consistent art style, readable text, no glaring visual errors. Using asset packs is fine, but mixing wildly different art without coherence can lose points under "art style fits the game." UI design that matches the theme and is easy to read (proper font choices, contrast, no tiny text) shows attention to detail – e.g. 2022 and onward explicitly gave points for adequate font size and contrast for accessibility.
- **Audio and Feedback** – Smaller but significant weight (5–10%). A game that has appropriate background music and sound effects for actions will outscore a silent game. Judges note if audio contributes to the game's mood/theme. In recent years they even folded this into presentation criteria (e.g. using the game's theme music or style in the slide deck).
- **Theme integration and Creativity** – This is a key differentiator. There are dedicated points for how well the game concept embraced the given theme. Judges look for a clear connection – if a team delivered a generic game that barely touches the theme, they will lose those points. Conversely, a clever or novel interpretation of the theme can impress. Creativity in mechanics and idea (something that stands out from expected solutions) is often rewarded in the design/presentation scoring. Many years explicitly list "innovation/originality of the game" as a criterion (worth ~2–5 points typically).
- **Presentation & Communication** – Roughly 15–20% of points consistently. This includes the slide deck content, the oral presentation, and the ability to explain design decisions. Trends here: judges expect **professional-quality slides** (no typo errors, not too much text, good visuals) and a confident delivery. Starting 2022, finals are presented in **English**, so fluency matters to some degree (clarity of speech is scored, though heavily accented English is okay as long as they can

be understood). Importantly, teams must **justify their design** – they need to articulate why their game is fun, how it meets the theme, how mechanics work, etc., showing they actually applied game design thinking. This reflective aspect grew in importance in recent years, as seen with references to MDA framework and dynamics/aesthetics explanation points.

- **Project management & process** – A smaller portion (perhaps ~5–10%) covers how teams managed the work. This may include the organization of project files, effective use of time (meeting deadlines for each module), and adherence to competition rules. In 2019 this was part of Module1 (planning worth 25p included project planning). In 2022 it was a tiny Module A (9p) mostly for project setup. By 2024, project management is likely evaluated more qualitatively (e.g. judges just note if the submissions were orderly and on time). Still, things like **missing a deadline** (not turning in a module on time) results in *zero* for that module – a devastating penalty. Thus time management is a silent but critical requirement across all years. All teams must stop when time is called; overtime work is not scored. This enforces that teams prioritize features and submit something playable by the cutoff each day.
- **Rising Complexity and Professionalism:** Over the years, the **expected quality bar has been raised**. Early on (2019) a team could win with a very basic game if it met requirements. By mid-2020s, criteria like **accessibility (colorblind checks, etc.)** and **design theory (MDA, target audience)** have been introduced. There's a clear push towards industry best practices:
- Using **inclusive design** (ensuring UI text is readable, color choices are accessible).
- Using formal **design frameworks** and being able to discuss them.
- Adhering to good coding standards (comment your code, organize your project logically).
- Treating the presentation like a business pitch to a client (entrepreneurial attitude in how the game is presented).
- Teams are effectively expected not just to code a game, but to function like a small indie studio producing a prototype for a client. The presence of a client-themed brief in 2024–25 underscores this.
- **What's Always *Not* Required:** It's equally telling to note what has **never** been asked for in any year:
- **Multiplayer or network features** – All games are single-player experiences. There's no requirement to handle networking or multiple simultaneous players, which would be too complex for a short competition. Even in genres like card games or strategy, the AI or challenge is computer-controlled, not another human.
- **Advanced AI or procedural generation** – While basic enemy behavior is expected, they never demand things like machine learning, complex pathfinding beyond maybe A* for enemies, or procedural level generation. If a team includes a rudimentary AI for enemies, that's usually enough. Creativity is welcomed, but no specific bonus for doing something like procedural content unless it directly enhances the game and you have time.
- **Complete games or large content** – Judges do **not** expect a full game with many levels or all assets polished. A single well-designed level or loop is sufficient. For instance, a tower defense with one functional level and a few enemy waves, or a card game with a limited deck and a couple of sample "recipes," is okay. Quantity of content is not scored – quality and functionality of whatever small slice you build is what matters.
- **High-end art or original assets** – There's no requirement that all art/audio be original. Using free or purchased assets is allowed (and common) as long as the team integrates them well and follows licensing. Judges care more about the cohesive use of assets than who created them. For example, a team can use asset-pack graphics and still score full points in art if those assets fit the game's theme and are used consistently. No points are explicitly given for making assets from scratch (though exceptional original art might subjectively impress judges).
- **Specific engine features** – The tasks never explicitly require using a particular engine feature or API. It's up to teams to decide how to implement things. There is also **no mandated engine version** given in the briefs we saw – but teams should use a stable version they know. As long as

the end product meets criteria (and runs on the provided PC), it doesn't matter if you used Unity, Unreal, Godot, etc. (Unity and Unreal just happen to be the recommended choices in documentation).

- **Teamwork scoring** – While it's a team competition in finals (pairs), there is no separate score for "how well the team worked together." Judges only see the output (game and presentation). So any internal task division is up to the team. A common pitfall, however, is poor division of labor – e.g. both team members working on code and neglecting art/sound, or vice versa. The best teams clearly split responsibilities to cover all aspects in time (implicitly improving their final product scores).
- **Common Pitfalls (What Loses Points):** Across all years, judges and briefs hint at several frequent mistakes:
- **Time mismanagement:** The most fatal error is not finishing a playable game by the deadline. An unfinished prototype (no win/lose condition, or crashes, or missing key mechanics) will sink the team's score in the largest module. Teams that overscope (try to implement too many features or a too-ambitious game) often run out of time. The judges would much rather see a **small but polished** game than an ambitious half-functional one. Missing the submission time for a module yields zero for that entire module – a fact stressed to scare teams into **scoping wisely**.
- **Weak theme incorporation:** If the game's connection to the given theme is superficial or unclear, teams lose the dedicated theme points and also risk a lukewarm reception overall. E.g. a team might build a generic platformer and just name the character to fit the theme – that's not enough. Judges expect the theme to manifest in gameplay mechanics or story meaningfully (as noted in multiple task descriptions and scoring rubrics). Not being able to clearly **justify how the game reflects the theme** in the presentation is a common point-loss area.
- **Poor presentation & communication:** Some technically strong teams fall short in how they present their work. Monotonic slides filled with text, spelling errors, or an incoherent explanation can cost easy points. For instance, a team might forget to include their game's **name or tagline** in the presentation, or fail to explain core rules due to nerves – these are avoidable issues. Practicing the final pitch and ensuring slides are clean can recover many points. Since finals require English, lack of preparation here can hurt (e.g. stumbling through the explanation without key vocabulary). The briefs explicitly mention aspects like text free of typos, readable fonts, and clear speech – seemingly small details that can add up.
- **Ignoring evaluation criteria:** Teams that don't pay attention to the judging criteria often miss out on free points. For example, if "color blindness check" is listed for 1 point, it's a trivial task to quickly test your game in grayscale mode or ensure high contrast. Similarly, failing to include a "tutorial or instructions" (worth 2 points almost every year) is a common oversight – some teams simply ran out of time to add a help screen, immediately losing those points. The top-performing teams typically hit every checklist item in the criteria, even if each is small, because together they constitute a significant score buffer.
- **Technical pitfalls:** These include forgetting to **package all files** (e.g. leaving out a DLL or an AssetBundle so the build the judges run is incomplete), last-minute **bugs** (a build that crashes on launch or has broken UI on the judging PC), or **performance problems** (e.g. uncapped frame-rate causing low-end PCs to lag, or physics glitches making the game unplayable). Many of these issues can be avoided by testing the build on the competition machine *before* submission. Not doing so is a risk.
- **Non-compliance with rules:** Using prohibited devices (e.g. a phone or external communication), or plagiarism would be disqualifying. While these issues are rare, the rules are clearly stated – no outside help, and all internet usage should be for searching existing resources only. Judges also keep an eye out for teams who might have prepared a base project in advance. If a game appears too polished or complex to have been made in the time (and doesn't obviously align

with the secret theme twist), suspicion can arise. However, teams are allowed to use code libraries or assets as long as they're available to everyone and properly credited.

Predicted Task Structure for Taitaja 2026 (Pelituotanto)

Based on the multi-year evidence above, **Taitaja 2026 Pelituotanto finals will almost certainly follow the 3-modular format**. The competition trend is towards realistic game development scenarios with a client brief and a strong emphasis on both technical and soft skills. Below is a detailed prediction of **all tasks (modules)** for the 2026 finals, including their purpose, likely requirements, and potential pitfalls:

Task 1: Concept & Pre-production (Module 1)

Core Objective: Conceive a game concept around a given **2026 theme** and produce a brief design plan. This is the foundation for the project – teams must understand the theme and outline how their game will implement it. The task will test the competitors' game design thinking and planning under time pressure. It also simulates receiving a **client's brief** and coming up with a pitch-able concept quickly.

Likely Theme & Format: The trend suggests a socially relevant or otherwise engaging theme. For instance, 2026 might have a brief like *"Client X wants a game prototype to educate or inform about [some topic]"* or a twist on a popular genre. Possible theme examples (hypothetical): **"Green Energy Tycoon"** (simulation game on renewable energy), or **"Mythical Maze Adventure"** (puzzle-platformer with mythology elements), etc. The exact theme is unpredictable, but it will be unique (not a repeat of previous years) and likely somewhat broad, allowing multiple interpretations. The brief will outline the target genre/audience loosely – similar to how 2025 specified a genre (card game) and theme (cooking). 2026 could combine a genre with a topic, e.g. *"an educational platformer for kids,"* or *"a strategy game about smart cities."* Expect the wording to be a **client request** ("The client is seeking a prototype of…") which signals the key goals.

Time Allocation: ~2–4 hours (likely closer to 3 hours) on Day 1. Recent years have shortened Task 1, implying 2026 will not spend more than half a day on planning. Teams will need to rapidly brainstorm and document their game idea in this window.

Mandatory Deliverables/Features:
- A **Design Document or Concept Presentation** drafted by the end of this module. This is usually a 1-3 page document (or a set of slides) covering: the game's **core concept**, how it addresses the theme, the **main gameplay mechanics**, and the artistic/style direction. It doesn't need to be extremely detailed (given the short time), but it must capture the *essence* of the game. For example, if the theme was "Green Energy Tycoon," the doc should describe what the player's goal is (build a renewable energy city?), what the main mechanics are (resource management, planning, etc.), and how that relates to the theme (teaching sustainability). Including one or two rough **concept art sketches** or diagrams (e.g. a level layout or UI mockup) could be expected, since previous tasks often mention concept art in presentations.
- A clear statement of the **game's genre and target audience**. Judges have been asking for target audience definitions (e.g. casual/mobile players vs. hardcore PC gamers?) as part of the design reasoning. Teams should specify who the game is for and on what platform (always PC for the prototype, but they might say "could be adapted to mobile" if relevant).
- **Inclusion of the secret theme in the concept.** The concept document must explicitly tie back to the theme. For instance, it should explain the narrative or gameplay elements that make the theme integral (not just pasted on). The 2026 brief will likely be revealed on the spot, so teams must show they've built their idea *around* it. Expect a requirement to mention "how the theme is utilized in gameplay" in the documentation (as was required in presentation criteria in prior years).

- **Project planning elements:** They may ask teams to list needed assets or a task breakdown for themselves. In 2022, a tiny part of scoring was project management. By 2026, they might explicitly prompt: "include a short work plan (who does what, or a timeline)". At minimum, teams should implicitly demonstrate planning (maybe a bullet list of features they will implement first, etc.). Judges look for indications of good planning, even if not separately scored – e.g. naming a game engine and outlining technical needs shows foresight.
- Possibly a **proof-of-concept or paper prototype element**: While not likely required, top teams often prepare quick prototypes even in this phase (for instance, a very rough movement test or mechanic test). The deliverable is usually just documentation, but any early prototype progress can be rolled into Module 2. The judges won't evaluate a prototype yet, but having one could de-risk the next stage. (This is just a strategy – the official requirement is the doc).

Evaluation Criteria & Points: Approximately **15–20%** of total points (expected ~15–20 points out of 100). Judges will score the design on creativity, completeness, and theme relevance. Key criteria likely: **Idea originality**, **adherence to theme**, **feasibility of the plan**, and basic **professionalism of the document** (clarity, organization, no typos). For example, was the concept interesting and not a direct clone of an existing game? Does it clearly address the client's ask? Is the planned scope realistic for a prototype? They will also note if the team considered things like target audience and user experience early. Since 2024/25 gave equal weight to design and presentation, we expect 2026 to value this module significantly – a well-thought-out concept can net full points here, giving a buffer going into development.

Things Unlikely to be Required in Task 1:
- **Detailed narrative or full storylines** – The design doc should outline the concept, but not write a whole game story. Time is too short; judges only expect a high-level concept.
- **Complete asset lists or technical specs** – While a brief plan of needed asset types is good, they won't need, say, a class diagram or full asset manifest at this stage. The planning is more conceptual than technical.
- **Prototype code** – No coding needs to be delivered yet (though teams might start setting up their Unity project). Judges will not look for a working prototype until Task 2.
- **Final art** – Rough sketches are fine, but no one expects final game art in the concept phase. Using placeholder drawings or even just verbal descriptions of art style ("pixel art 2D" vs "3D low-poly") is sufficient.
- **Long documentation** – Brevity is actually encouraged. A concise concept that hits all points is better than a verbose doc no one has time to read thoroughly. Likely a one-page concept summary or a short slide deck is all that's needed. Overly long design docs could even be counterproductive given the time wasted on them.

Common Failure Risks in Task 1:
- *Misunderstanding the Theme:* Some teams might pitch a game that only superficially matches the theme or, worse, ignores key client requests. This will set them back not only in Module 1 scoring but for the whole project. To avoid this, teams must **identify the core keywords in the brief and ensure their idea addresses each one**. Judges have penalized teams who had to retro-fit the theme later.
- *Over-scoping the Concept:* A design that sounds awesome but is clearly too ambitious for a 2-day build is a trap. Judges can sniff out when a plan is unrealistic. If a team proposes a massive open-world RPG or dozens of levels, it will raise eyebrows. The best concepts are *focused* – they have a clear core mechanic that can be prototyped quickly. Over-scoping in the design phase also makes it harder to prioritize during development (which can lead to an unfinished game in Task 2).
- *Poor Team Collaboration:* In this phase, both teammates need to contribute ideas and align on a plan. A common pitfall is one teammate dominating the concept creation without buy-in from the other, leading to execution problems later. Conversely, losing an hour due to disagreements or "design by

committee" indecision can eat into an already short module. Teams should practice rapid brainstorming and consensus-building to avoid this.
- *Neglecting the Design Document Quality:* Some competitors might think "it's just the idea, the real work is coding" and jot down a very hasty, disorganized concept note. This can lose easy points. Sloppy or minimal documentation (e.g. a few bullet points that don't cover all requested aspects) will score poorly. It also serves as a weak blueprint for Task 2, increasing the chance of overlooking requirements.
- *Missing the submission deadline:* It sounds absurd to miss a deadline so early, but it has happened – e.g. teams getting carried away and not exporting their concept doc in time. Given Module 1 is short, there's zero slack. A team that doesn't stop to submit (e.g. still writing when time is called) risks getting 0 for the whole module. The rules make it clear: if you don't submit by the cutoff, that module isn't evaluated. So time management and knowing when to wrap up the design is crucial. Setting an alarm 10 minutes before the deadline to finalize and save the document can mitigate this risk.

Task 2: Prototype Development (Module 2)

Core Objective: Develop a **working game prototype** that implements the core gameplay envisioned in Task 1, within the constraints of the theme and competition timeframe. This is by far the largest and most critical task – where coding, asset creation, and integration happen. It tests the competitors' technical game development skills, their ability to divide workload, and their discipline in delivering a playable game under time pressure.

Expected Scope and Complexity: For 2026, we anticipate roughly **12–14 hours** of development time split across Day 1 (afternoon) and Day 2 (all day). This is consistent with recent competitions (which offered ~13–14h for development). The game should be a **single-level or single-session prototype** demonstrating all key mechanics, rather than a full game with multiple extensive levels. Complexity should be moderate: enough to be interesting and show skill, but not so much that it can't be finished. For example, if the genre is strategy, one scenario or mission is enough; if it's a platformer, a single well-designed level is fine; if it's a tycoon sim, a small set of systems with a win condition (like reach X money) would do.

Mandatory Mechanics & Features: (These will be explicitly or implicitly required based on pattern – the judges' checklist for the game build tends to be very similar each year.)
- **Functional Core Mechanics:** The primary gameplay mechanic(s) described in the concept **must be implemented and working**. If the game is about puzzle-solving, there must be puzzles that can be solved. If it's a deck-builder, the deck must shuffle and deal cards correctly. Essentially, the "promise" of the concept needs to manifest in playable form. This is usually worth significant points – judges will check if "Mechanics work as intended". Expect around 8–10% of points just for core mechanics functionality.
- **Player Controls and Camera:** The player (or player's agent in the game) should be easy to control. Whether it's character movement with keyboard/controller or navigating menus with mouse, the controls must be **intuitive and responsive**. Any physics (jumping, shooting, etc.) should feel reasonably tuned (not floaty or unresponsive). The camera (in a 2D or 3D game) must properly show the gameplay without awkward angles or clipping – e.g. following the player smoothly in a platformer, or offering the right zoom level in a strategy view. If 2026's game is 2D, camera control might be minimal (just ensure the viewport covers action). In 3D, implementing a basic follow or orbit camera is expected if relevant.
- **Clear Win/Lose Conditions or Goals:** The prototype needs a **game loop** – i.e. a situation where the player can succeed or fail. Judges will look for a win condition (score reaches X, boss defeated, puzzle solved, etc.) or an endless high-score mechanic, and a fail state (player dies, runs out of time/resources, etc.). The game doesn't necessarily end and close, but it should communicate to the player that they've won or lost when it happens. For example, a message, or transition to a summary screen. In prior years, having a fun gameplay loop was explicitly scored. This means teams must not forget to implement the

end states and any scoring system – even a simple text "Game Over – you survived 5 waves!" is better than nothing.

- **User Interface & Feedback:** The game must provide feedback to the player through a UI. **HUD elements** like health bars, score counters, resource meters, etc., appropriate to the game type, should be present. At the very least, a **score or progress indicator** is expected if applicable (for instance, a timer for a timed challenge, a score for an endless game, or a level completion percentage). Additionally, the UI should include **player instructions and controls** – often delivered via a start screen, pop-up, or an in-game help menu. Judges award points for "Build contains all relevant gameplay instructions", so the team must include a quick guide (e.g. WASD to move, objective: reach the end, etc.). Given increased emphasis on accessibility, UI text should be of adequate size and contrast (teams can easily meet this by using high-contrast colors for text and a readable font – which they should plan for in Task 1). Expect at least a couple points for having a **clear and visually appealing UI**. Thus, UI design cannot be an afterthought – it should be functional and on-theme (e.g. if the theme is mythical, maybe the UI has a fitting font or frame graphics, though readability comes first).
- **Graphics and Art Implementation:** By the end of Task 2, the game should have a complete set of visuals for a prototype. This includes a **playable environment** (tileset or background for 2D, or a simple 3D level), **characters or game pieces** (player character sprite/model, enemy sprites/models, cards images, etc.), and UI assets (buttons, icons). The art does not have to be original – teams can use pre-made assets – but it needs to look **cohesive and appropriate to the theme**. Judges will check if the art style fits the game's concept (no mishmash of realistic and cartoon assets, for example). If the brief was client-oriented, the expectation is that the "client's theme" reflects in the art: e.g. in a sustainable energy game, visuals might include solar panels, green landscapes, etc. We predict an explicit scoring item like "Art style is appealing and supports the theme" worth ~5–6 points, as seen previously. - **2D vs 3D:** It will be up to the team. Most likely the majority will choose **Unity 2D** because it's faster for small teams. The judges do not mandate one or the other. If a team goes 3D, they must be able to produce or import 3D models quickly (note: common strategy is to use simple shapes or free models to save time). Both approaches can score full points if done well. The key is polish in whichever dimension: a well-made 2D game can beat a clunky 3D game. Unity (2020+ versions) and Unreal both are available; Unity 2D has been the go-to for many past winners, so focusing on 2D proficiency is a safe bet unless the theme really calls for 3D (which is rare).
- **Audio:** The prototype is expected to include **background music and sound effects**. By 2026, it's standard that games have at least one music track (even if looped) and sounds for main actions (jump, shoot, card flip, etc.). Teams should prepare to quickly find or create a few sound effects and a fitting music loop. Judges will specifically listen for how well the audio matches the game atmosphere and theme. For example, a cooking card game might have playful, upbeat kitchen music and chopping sound effects; a mythic adventure might have epic or ethereal music and appropriate ambient sounds. Audio is often easy points if not neglected – maybe 5% of the score. So in 2026, teams must budget a little time for implementing audio (dragging in sound files to triggers). Silence or random unfitting audio will lose those points.
- **Technical Performance:** The game must run **smoothly** on the provided competition PCs. Typically, competition machines are mid-range; any unoptimized game that causes frame drops will lose a point or two. In practice, that means avoid doing extremely heavy computations or excessive physics, and be mindful of things like particle systems or unbounded loops. For instance, capping the frame rate or using Unity's built-in profiler to ensure there are no memory leaks can help. Also, a common oversight is not disabling vsync or heavy post-processing from templates – teams should check these if they use engine defaults. The 2026 criteria likely include "Game runs without performance issues on test hardware" as a checklist item (usually 1–2 points).
- **Accessibility & Settings:** Building on recent trends, 2026 may require basic accessibility considerations. This could mean: ensuring color choices are distinguishable for colorblind players (maybe they'll test by toggling a grayscale filter), having readable fonts (no tiny illegible text), and possibly an option to adjust volume or other simple setting. While it's unlikely they demand a full

options menu (time is limited), the criteria might specifically mention “game UI passes contrast checks” or similar. The safest assumption is teams should use high-contrast UI design and avoid relying on color alone to convey information (e.g. use symbols or labels as well). Given it’s been explicitly scored before, teams that address this (like including a colorblind mode toggle or choosing a colorblind-friendly palette from the start) might earn bonus kudos.

Engine and Tools Assumptions: The competition will provide the latest stable versions of Unity and Unreal (and possibly alternatives like Godot) on the PCs, but **Unity (2D)** is expected to remain the optimal choice for most. It provides the quickest pipeline for both programming and asset integration for small scope games. All years’ documents reference Unity/Unreal, indicating those are the intended tools. Unity’s familiarity in previous Taitaja events and its Asset Store access for quick asset grabbing make it invaluable. A Unity 2021–2022 LTS or Unity 2023 version will likely be installed; competitors should verify and practice with the version to avoid surprises (project upgrade issues, etc.). If a team is exceptionally skilled in Unreal (especially for 3D games), they could use it, but they must weigh slower iteration times. C# in Unity is generally quicker for small mechanics than C++/Blueprint in Unreal unless the team’s project specifically benefits from Unreal’s strengths (e.g. superior 3D visuals out-of-the-box). In short, **focusing on Unity 2D is still the optimal strategy** for a wide range of possible themes – it allows rapid prototyping of common game types (platformers, top-down games, puzzle interfaces, etc.) with minimal friction. Teams should prepare a base Unity project with packages they might need (2D tools, Cinemachine for camera, TextMeshPro for UI text, etc.) to save setup time in competition.

Evaluation Criteria & Points: This module will carry the bulk of points – expect **50–60%** of total (likely 55–60 points out of 100, if following recent patterns). The criteria will be similar to 2022–2025’s “Game Build” category, covering: Gameplay (fun & functioning), Graphics (art quality/fit), Audio, Technical execution, perhaps an **Innovation** item (did the team add any extra creative features beyond the basics?), and possibly **Code quality/organization** in some form. Judges might not inspect code deeply, but they often have a line in the description about “project is well structured and code is commented” – which was more of a note than points in 2022. Still, teams should behave as if code could be reviewed, meaning keep it relatively clean and commented especially if using others’ code.

One thing to note: **WorldSkills influence** – if Pelituotanto is aligning more with WorldSkills, the judging might involve aspect-based scoring (like points tied to WSOS sections: e.g. Game Design, Game Art, Programming, etc.). We already see signs of that (e.g. “WorldSkills Occupational Standards” mention in 2024 scoring forms). So in 2026, points might be grouped by broader skill areas, but effectively they map to the same specifics (e.g. “Game functionality” covers mechanics/coding, “Artwork” covers visuals/UI, “Game design” covers how well the concept plays and theme usage, etc.). Competitors should ensure to cover all skill areas enough to not leave any section blank – a very polished mechanic with no audio or poor presentation still won’t win if it zeros out an entire category.

Likely Pitfalls in Task 2: (This is the stage where most competitions are won or lost, so many pitfalls exist – we list the common and the 2026-specific ones to watch out for.)
- *Scope Creep & Feature Blowout:* Perhaps the #1 risk. Even with a good plan, teams often get ideas during development (“Wouldn’t it be cool if we also added X?”) and chase them, consuming time needed for core features. In 2026, with an intense schedule, teams must practice discipline: stick to the **Minimum Viable Game** first. Implement the absolutely essential mechanics before adding any “nice-to-haves.” A strategy is to list features as *Must*, *Should*, *Could* – if Musts aren’t done, do not start Shoulds. Overscoping leads to prototypes with missing pieces (e.g. an inventory system half-implemented, or an additional level that’s unfinished while core gameplay is buggy). This will be heavily penalized, as incomplete features can break the game.
- *Integration Bugs:* When splitting tasks, one teammate might work on mechanics while another works on UI or art, etc. Merging these can cause issues (e.g. the mechanic code isn’t linked to the UI properly).

Lack of integration testing throughout Day 2 can result in a final hour scramble where nothing quite works together. To avoid this, teams should frequently build and test the game as they add features – catching bugs early. A typical pitfall is leaving UI implementation last and then finding out the numbers don't display or buttons don't trigger events minutes before deadline. The better approach is iterative: get a basic playable loop early on Day 2 (even with greybox graphics), then refine.

- *Ignoring Playability/Balance:* Sometimes teams implement all features but never tune the gameplay. For example, they might have an enemy that is so fast it kills the player instantly, or puzzles that are unsolvable. Judges will play the game; if it's ridiculously hard, or conversely not challenging at all, it detracts from the "fun gameplay loop" points. Competitors should playtest their own game a few times and adjust parameters (speed, health, spawn rates, etc.) to ensure the game is reasonably balanced and *fun for a first-time player*. Since judges typically only spend a few minutes on each game, the first impression matters – it should be neither frustrating nor trivial.
- *Lack of Feedback to Player:* A common issue is when actions in the game have no feedback – e.g. the player clicks a button and there's no sound or visual cue, an enemy hits the player and there's no flash or health decrement shown. This makes the game feel unpolished and can confuse judges (they might not even realize something happened). The criteria indirectly cover this in UI/visual clarity and audio. Teams should ensure every important action has some feedback: sound effect, particle, screen shake, UI update, etc. It doesn't have to be fancy, just noticeable. Missing feedback loops can also cause scoring deductions under "gameplay mechanics create a good experience". For instance, defeating an enemy should maybe play a sound and increase score – if it just disappears with no fanfare, the moment feels flat. These small touches often separate the top projects.
- *Crashes and Freezes:* Any crash will likely cost a team dearly. Whether it's an outright crash-to-desktop or getting stuck in an unwinnable state, it prevents judges from fully evaluating the game. Typical culprits: null reference exceptions (like trying to play a sound that isn't assigned, etc.), uncaught errors, or build issues. Also, using experimental engine features or unstable packages can introduce crashes. Teams should use well-tested techniques and, importantly, **test the final build** on the competition PC before submission. Running the .exe (not just in Unity editor) can reveal issues like missing files or case-sensitivity bugs. Given the one-chance nature, it's wise to allocate time (at least 15–30 minutes before final deadline) to build the game and do a quick run-through of the packaged version. If any crash is found, even if it means cutting a feature last-minute, it's better than submitting a broken build.
- *Performance Neglect:* While simple prototypes usually run fine, it's possible to inadvertently bog down performance (e.g. spawning thousands of particles, using very high-poly models or uncompressed textures, or an inefficient algorithm in Update()). A game that lags or stutters will lose the "smooth performance" points and also sour the gameplay experience. Teams should be mindful of this – for example, limit physics objects, use object pooling for bullets/enemies, and avoid expensive operations in tight loops. If using Unity, enabling VSync or setting TargetFrameRate can prevent it from overusing CPU/GPU. For 2D games, large transparent textures can cause draw calls that slow things – slicing sprites or using sprite atlases could help. While micro-optimizations aren't the focus in a 2-day build, **basic performance hygiene** is expected. If judges see a game noticeably chugging, they'll assume the team lacked optimization knowledge. In 2026, with increased awareness of optimization (often part of WorldSkills), a smoothly running game reflects well on the team's skill.
- *Asset Overload or Art-Integration Issues:* Teams might grab lots of assets from the internet (because it's allowed and can save time). But dumping too many uncoordinated assets can lead to a visually incoherent or technically problematic game. Problems include: textures that bloat the project (leading to slow load times or large file size), models that require tweaks (colliders, scaling), or mixing art styles that look jarring. The pitfall is spending *too much* time hunting/downloading assets and not enough integrating them properly. A smart approach is to limit asset sources: choose one art pack for environments, one character style, etc., to maintain consistency. And import only what you need. Additionally, crediting assets (maybe in a text file or in-game credits screen) is wise – while not explicitly scored, it's good practice and proves you're not claiming them as your own. Judges likely won't penalize lack of credits, but given the professionalism push, including a brief credits note can't hurt.

- *Unbalanced Workload between Team Members:* In a pair, if one person is coding everything and the other is idle or only making art, that can bottleneck progress. Conversely, two people working in the same code without coordination can cause merge conflicts or wasted effort. Many teams falter by not clearly dividing tasks. The ideal pattern (observed in successful teams) is **parallel work**: e.g., one handles player & mechanics programming, the other sets up levels, designs UI, and imports assets, then they integrate. In 2026, teams that have pre-defined roles ("I'll handle programming X and Y, you handle art and UI") and can work simultaneously will maximize productivity. If one member has weaker coding skills, they can take charge of level design, testing, documentation, etc., to ensure the game gets polished. A pitfall is failing to leverage both members fully – effectively losing manpower. Practicing collaboration and maybe using version control (if allowed offline) or at least careful syncing via Unity Collab or manual file exchange is important so they don't overwrite each other's work.
- *Forgetting Module Submission:* Similar to Module 1, but an even bigger disaster here – not exporting the final build or not copying it to the submission folder by the deadline. With exhaustion setting in, a team might think "just another quick tweak" at the last second and then time is up. That would mean zero points for the biggest module. To avoid this, teams should target to have a playable build slightly before the deadline (even if missing one minor feature) and submit that. If there's time for a minor improvement, fine, but they should *never jeopardize having nothing to submit*. A good practice is creating a "final build" 15 minutes early. If a catastrophic bug is discovered, at least an earlier working build is there to turn in.

Task 3: Final Presentation & Post-mortem (Module 3)

Core Objective: Present the game to the judges (and possibly an audience), demonstrating its features, explaining how it meets the given theme and client's goals, and reflecting on the design and development. This task evaluates communication skills, understanding of the project's strengths/weaknesses, and how well the team can sell their idea. It's effectively the "post-production" phase where the team packages their work into a convincing pitch.

Format: Day 3 of the competition is typically for presentations. In 2026, expect the morning dedicated to final prep (likely ~2 hours to polish slides, rehearse, and ensure the game is ready to demo) and then the **presentations themselves in front of judges** (each team usually gets around 10–15 minutes: e.g. a 5–7 minute presentation plus Q&A or demo time). The exact format can vary, but often it's a stage presentation with a projector. Teams will speak in **English** (as mandated since 2022) to simulate international settings. This module's submission usually includes the slide deck (e.g. PowerPoint file) by the deadline, and then the live presentation event is evaluated. Judges have the slides and possibly the game build to follow along as needed.

Mandatory Presentation Content:
- **Introduction of the Game:** Teams should start by clearly stating the game's title and tagline, and give a quick overview of what type of game it is. Judges want to immediately catch what the concept is (e.g. "*Our game Solar City Tycoon is a single-player strategy simulation where you manage a city's energy grid using renewable resources.*"). A catchy name and tagline that encapsulate the theme are often specifically awarded points – expect 1 point for a good name/tagline that show "entrepreneurial attitude." Thus, teams should come prepared with a decent name (not something like "Game2026").
- **Theme Integration Explanation:** Early in the presentation, competitors must address how their game fulfills the **given theme or client problem**. This was a big chunk of points in previous years (e.g. 10 points for explaining use of theme in 2022). In 2026, they should explicitly mention the theme and demonstrate with examples from their game. For instance, *"The theme of sustainable energy is reflected in our core mechanics: players must balance solar, wind, and hydro power to keep the city running, which directly teaches the challenge of renewable energy storage."* Essentially, they need to convince judges that the game wasn't just generically made – it's a direct response to the prompt. They can show screenshots

or a short live demo clip highlighting these themed elements.
- **Core Mechanics & Gameplay Summary:** The team should describe **how the game is played** and what makes it fun. This is explaining the **Mechanics and Dynamics** – what the player can do and what experience that creates. They might walk through a typical play session or level: "First, you do X, then Y happens, your goal is to achieve Z while handling obstacle Q," etc. Judges give points for clearly explaining the **core game mechanics** (worth ~5 points in past) and the **game dynamics** (the behavior/ emotions that arise from those mechanics, worth a smaller amount). For example, "Our card game's core mechanics are deck-building and hand management – the dynamic it creates is a tense resource trade-off where players must decide which ingredient cards to use or save for later, creating strategic depth." This shows understanding of design.
- **Innovation & Uniqueness:** Teams should point out what is **unique or innovative** about their game. Even if the game is based on an existing genre, what twist did they add? Maybe it's the theme integration itself, or a novel mechanic, or a creative art style. Judges often allocate a couple points for innovation/originality. Teams that explicitly call it out ("Unlike typical tower defense games, in our game the towers themselves move because they are magical creatures – this twist adds unpredictability and strategy.") signal to judges that they aimed for originality. If the team had to cut a planned innovative feature due to time, they could mention it hypothetically, but carefully (only if they still did something different to talk about; you don't want to highlight what's missing, but can say "Originally, we planned feature X to further enhance Y, which could be a future development avenue." This shows foresight without dwelling on incomplete aspects).
- **Aesthetic and Audio Design:** A portion of the presentation should cover the **Aesthetics** – both visual and audio – of the game. Teams can show art assets they created or chose, and justify why that style was chosen (tying back to theme or audience). For instance, "We opted for a colorful low-poly art style to appeal to younger audiences and to keep performance smooth." If they had to make interesting art or audio decisions, mention them. Also, referencing that they considered accessibility (like "We chose a dyslexia-friendly font and ensured all UI text has high contrast") can impress judges that they went the extra mile, even if not explicitly asked. Judges might not have separate points for this in presentation, but it reinforces criteria from Module 2 in their minds and shows holistic thinking.
- **Development Process & Challenges (Post-mortem):** Many judges appreciate when teams briefly reflect on **how the development went**, and any challenges overcome. It's not always formally scored, but it demonstrates professionalism to acknowledge what was hard and how it was handled. For example, "One challenge was balancing the game within the time – we realized early on our original level was too easy, so we tweaked enemy spawn rates to increase difficulty by the final presentation." Keep this section brief and spin challenges as learning experiences or smart adjustments. If a feature had to be cut due to time, it can be mentioned in a forward-looking way ("Due to time, feature X is not fully implemented, but we have laid the groundwork and would include it in a full version"). This shows scope management rather than failure. Some competitions give a point or two for "realistic assessment" or similar. Regardless, it helps humanize the project and let judges see the team's problem-solving attitude.
- **Conclusion and Future Vision:** End the presentation with a concise summary of why the game meets the client's needs and possibly what could come next. E.g. "In conclusion, **Solar City Tycoon** successfully engages players with the sustainable energy theme in a fun simulation. With more time, we would add more energy sources and scenarios, but our prototype proves the core concept. Thank you." This wraps it up confidently. Mentioning future expansion ideas (briefly) shows that the team knows the prototype's limits but sees its potential – which is basically what a client would want to hear in a pitch. It's a nice way to address any missing polish indirectly ("we know what else we'd do given more time").

Presentation Delivery Expectations:
- Teams must present in **English**, so fluent or well-practiced delivery is essential. Clarity of speech is scored, so teams should rehearse what they will say to avoid stumbling, and ensure they pronounce game-specific terms clearly. A heavy accent is fine as long as judges can follow; using simple language

is often better than reading a complex script awkwardly.
- **Slide Design:** The slide deck should be visually appealing and not text-dense. Judges give points for slides that are easy to read (suitable font size, good color contrast, etc.) and not overfilled (they had a criterion "amount of text per slide is manageable" – implying use bullet points or short phrases rather than paragraphs). Including images – screenshots of the game, concept art, diagrams – is highly encouraged, as it makes the presentation engaging and also proves the points being made. The slides should incorporate the game's visual identity – e.g. use the same color palette as the game or similar fonts for headings (this was explicitly scored before: using the designed game color palette in the presentation). Essentially, treat the slides like a marketing pitch deck for the game.
- **Demonstration:** Usually teams will **live demo the game** either during or right after the slide presentation. It might not be explicitly required in text, but practically judges often want to see it in action. Teams should coordinate who will talk and who will control the game if doing a demo simultaneously. It's wise to have a pre-recorded short gameplay video embedded in slides as backup (in case a live demo fails or time is short). This ensures the judges see the game's best moments without technical hiccups. However, live demo can be impressive if the game runs well – it shows confidence. If doing live, plan a scenario that shows off the features quickly (e.g., use cheat keys to jump to a later level or trigger certain events if needed so you don't waste time). Also be prepared to answer judges' questions after – common questions: "How did you implement X mechanic?" or "Why did you decide to do Y?" – basically checking that the team really did the work and understands it.
- **Time Management:** Presentations are typically time-capped. A pitfall is trying to cram too much and then being cut off. Teams should aim to fit in the allotted time (practice to know your timing). Hitting all the key points succinctly is better than going on a tangent. Judges will have read the slides as well; don't just read them verbatim – add value with what you say (explain bullet points, give examples). If time ends and a team hasn't mentioned theme or core mechanic, that's a big fail. So prioritize those early in the talk.

Evaluation Criteria & Points: Expect roughly **20–25%** of total points allocated to Module 3 (likely ~20 points as per 20/60/20 splits). Specific points breakdown as evidenced in prior years includes: Slide quality (visuals, readability) ~5 pts, Speech clarity/confidence ~1–2 pts, Game concept explanation (mechanics, dynamics) ~5+ pts, Theme explanation ~5+ pts, Originality/creativity highlighted ~2 pts, maybe target audience & intended experience ~1–2 pts. Also, overall coherence and professionalism of the presentation will influence scoring – judges have discretion if a presentation really wowed them, even if not explicitly in rubric, it can reflect in how they assign subjective points for criteria like "explains game design (10 pts)" – they might give 9–10 for an excellent pitch vs 5 for a confusing one. Essentially, a well-delivered presentation can solidify the impression of a great project or help redeem a project that had minor shortcomings by framing it well. Conversely, a poor presentation can undermine a great game (if judges are left unsure what was accomplished or why certain decisions were made). Competitors should recognize Module 3 is their last chance to influence the judging – it's not just fluff, it can swing the ranking especially if teams are close after Module 2.

Likely Pitfalls in Task 3:
- *Lack of Preparation/Practice:* A common mistake is teams spend 100% of time on the game and 0% rehearsing the presentation. They then fumble through slides, talk in circles, or run out of time during the actual pitch. This looks unprofessional and will cost points even if the game itself is strong. In 2026, teams should allocate a bit of Day 2 evening or Day 3 morning to practice what each person will say. Not doing at least one full run-through is a risk. It's easy to tell who practiced: their delivery is smoother and timing on point.
- *Overloading Slides with Text:* Despite criteria telling them not to, some teams will put their entire design essay on slides. This will definitely lose the "manageable text" point and also makes it hard for judges to glean the key info. Another aspect is font size – using tiny text to fit more in will lose the readability point. Pitfall: forgetting that slides are often viewed from a distance or by multiple people – small or

dense text simply won't be read. The remedy is using keywords and images. Judges have the design doc for detail; slides are for highlights and persuasion.
- *Ignoring Aesthetics in Presentation:* If the presentation looks mismatched or ugly compared to the game, it's a missed opportunity. For example, using a default PowerPoint template with corporate graphics when your game is a colorful fantasy – that disconnect can subtly weaken the impact. Some teams might ignore the instruction to use the game's visual style in slides. That's an easy fix: use a background or border from your game art, or at least the same color scheme. Not doing so could lose that specific point and generally feel less cohesive.
- *Speaking Issues:* These include reading directly from notes or slides (with little eye contact or engagement), speaking too quietly or too fast (especially if nervous or not fluent in English), or splitting speaking roles awkwardly (one person speaks 90% while the other just clicks slides – making judges wonder if it was a one-person effort). Teams should plan equal or at least meaningful participation in speaking for both members. If one's English is stronger, that person can take more, but the other should at least present something (like the technical parts or the art parts) to show both contributed. Another issue is Q&A: being unprepared for judges' questions. Teams might freeze or give contradictory answers (if they haven't discussed possible questions). To avoid this, consider likely queries ("How did you implement X?", "What would you improve?", "Why did you choose this engine/technique?") and agree on responses. Not being able to answer about a part of your game (say a teammate coded it and you have no idea how it works) can signal lack of understanding or teamwork.
- *Technical Difficulties:* We've seen things like videos not playing, wrong version of slides, or the game failing during live demo. These can derail a presentation. Teams should **test their presentation setup** – if they plan to demo, have a backup plan (video or screenshots). If using a video, make sure it's embedded properly or have it open in a player ready to switch to. If slides were made in an uncommon software, export to a common format (PowerPoint or PDF) to avoid incompatibility on stage. Little things like bringing their own clicker or ensuring the display resolution works with their slides (no cut-off edges) can avoid hiccups. While minor technical issues might not directly cost points (judges can be forgiving if handled gracefully), they eat into your precious presentation time and can break your flow, causing you to rush or skip important points.
- *Overemphasizing or Underemphasizing the Right Things:* Some teams spend too long on backstory or minor features and not enough on core mechanics and theme – misallocating their presentation time. Judges will score what they asked for: theme explanation, mechanics, etc. If a team uses 3 minutes on a lore backstory or showing every menu screen and then runs out of time to explain how the game actually plays or how it meets the theme, they'll lose points. Conversely, a presentation that is too dry and technical, not showing any excitement or visuals, can fail to engage. The pitfall is forgetting this is **also a sales pitch** for your game. Judges are human – a bit of showmanship (an enthusiastic tone, a short exciting demo video with music, etc.) can leave a strong positive impression. Teams that treat the presentation as merely a formality often miss that it can differentiate them. In 2026, with likely several teams all having somewhat working games, the presentation can be the tiebreaker. So the content needs to hit all rubric points, but also be delivered with confidence and a spark of passion. Showing that the team really believes in their game can influence subjective scoring for things like "game design explained" or "innovation" because excitement is contagious.

Preparation Guidance for Competitors Heading into 2026

To excel in Taitaja 2026 Pelituotanto, competitors should start preparing **well in advance**. Below is a comprehensive checklist and advice, distilled from the patterns and requirements observed over the past years:

Technical Skills & Project Setup

- **Game Engine Mastery (Unity 2D recommended):** Make sure you are **highly comfortable with Unity** (especially 2D features, if you plan to go 2D). This includes: setting up new projects quickly, importing assets, using the prefab system, scene management, UI canvas system, and building executables. Practice creating small projects from scratch so the initial setup is second nature. If you prefer Unreal, ensure you're adept with Blueprints/C++ for basic gameplay – but note that Unity is generally faster for 2D and UI-heavy games. Unity's Asset Store also gives a quick content advantage (e.g. free pixel art packs, sounds) which can save time, so being familiar with integrating Asset Store packages is useful.
- **Core Mechanics Coding:** Prepare template or reusable code for **common mechanics**. For example:
- Character **movement and physics** (platformer controller, top-down movement, etc.). You should have a script ready for smooth WASD or arrow key movement, jump with gravity, etc., which you can adapt to any game that needs character control.
- **Camera follow/control:** Know how to set up a camera that follows the player or one that scrolls a level. In Unity, learn Cinemachine – it's excellent for quickly getting a nice camera without coding from scratch.
- **Enemy AI or hazard behavior:** Write a simple state-based enemy script (patrol-chase-attack) or a spawn-and-move pattern (like enemies that move towards target or along a path). This can cover many genres (tower defense enemies, platformer monsters, etc.). Also know how to do simple physics triggers/collisions (so you can handle things like bullets hitting enemies, player hitting obstacle, etc.).
- **Collectibles & Scoring:** Be ready to implement item pickups, score increments, timers, health system, etc. It helps to have a modular system (maybe a GameManager script that keeps track of score, time, lives and UI updates accordingly). Many games need one or more of these – having a ready pattern will save time.
- **Menu and UI Screens:** Practice making a start menu, pause menu, and game over screen. Know how to tie button events to functions (e.g. Start Game, Quit, Resume) in your engine of choice. Set up a basic UI canvas with a few buttons and text elements quickly. This preparation ensures you can meet the UI requirement (instructions to player, start/restart controls) with minimal effort. A pre-made Unity scene for a start menu (with placeholders) that you can import and tweak might be a good idea.
- **Audio Management:** Have a straightforward method to play music and sound effects. For example, in Unity, know how to use AudioSource for one-shot sounds and looping background music. Perhaps write a small AudioManager that can be fed a list of clips and handle playing them (with volume control). This way, adding audio is just dropping files in and calling a play function.

- **Reusable Utilities:** Little scripts like a Timer, Object Pooler (for reusing bullets/enemies), or a simple Finite State Machine template for AI can be pre-written and practiced. While you must write code during the competition, being intimately familiar with these patterns means you can recreate them from memory quickly. (It's not against rules to remember code – it's expected. Just no copy-paste from outside.)

- **Asset Integration:** Practice quickly **importing and adapting assets**. For example, take a random sprite sheet and implement an animation in Unity, or import a 3D model and adjust its collider and materials. Also practice using Unity's Tilemap if 2D tile-based levels are likely (for platformers or top-down). If you expect possibly doing a 3D game, practice basic Blender usage or at least adjusting existing models (scaling, adding colliders in Unity). Knowing how to do

lightmapping or post-processing is less crucial than simply getting assets game-ready (e.g. no pink textures, correct orientation, etc.).

- **Optimization Basics:** Be aware of simple performance tricks: use appropriate update loops (e.g. don't put heavy code in Update if it can be event-driven), destroy or pool objects that are no longer needed, avoid extremely high-polygon assets or huge textures, and know how to profile in Unity if something is lagging. While top performance isn't the goal, avoiding obvious slowdowns is necessary for that smoothness point.
- **Version Control or Backup Plan:** Although internet is off-limits, you can use offline Git or collaborate through Unity Collab if set up prior (or simply manual sync). The key is to have a **workflow for both team members to work simultaneously without overwriting work**. Practice this! For instance, split scenes or split prefabs: one person works on Level1 scene, the other on UI and then merge by adding the UI canvas prefab to Level1, etc. If using Git offline, practice making commits and merging on your own beforehand to avoid repository mishaps. At minimum, frequently back up the project folder (copy-paste) especially before risky changes – this can save you if something corrupts.
- **Hardware Familiarity:** If possible, practice on a similar setup to what the competition offers (typically Windows PCs). Ensure you know all the **keyboard shortcuts** and settings in your tools to move fast. For example, in Unity: duplicate objects with Ctrl+D, navigate the scene view quickly, use the animation editor, etc. Set up a custom layout that you like for the engine and remember it (you can quickly configure it on-site). Little things like these save seconds that add up over 14 hours. Also, if you have custom Editor tools or assets (that are allowed and don't violate any rules), prepare them. E.g., some teams bring a personal package of favorite scripts or shaders – if the rules allow local assets, that can be a boost. Just make sure they truly don't count as outside help (if they are your own from scratch and available to all team members, it should be fine).

Game Design & Thematic Knowledge

- **Study Past Themes and Genres:** Review all past finals (which you've done through these materials) and also think of other classic competition game themes (common ones in game jams: space exploration, ancient myths, survival, etc.). **Brainstorm game ideas for each** – it's a great exercise. The goal isn't to guess the exact 2026 theme, but to train flexibility. If you've already ideated a half dozen distinct game concepts for different themes, you'll be faster when a new theme is revealed. You'll also notice patterns: many themes boil down to certain game types. For example, educational or awareness themes (like sustainability) lend themselves to simulation or puzzle mechanics; action themes lend to platformers or shooters. Practice coming up with a unique twist for common genres (what could be a twist on tower defense? On a quiz game? On a farming sim?). This mental library of ideas can inspire your actual concept under pressure.
- **Game Mechanics "Toolbox":** Make sure you are familiar with a variety of game mechanics and can mix and match. Know the basics of platformers (moving platforms, spikes, power-ups), shooters (projectiles, enemy spawn patterns), puzzle games (switches, keys and doors logic), strategy (resource management, AI opponents), and so on. You don't need to master making all of these, but understanding how they work means if the theme leans that way, you won't be starting from zero. For instance, if 2026 theme ended up being a management sim, you should know the concept of cycles, resource production and usage, upgrades, etc., so you can implement a simplified version. If it's an arcade action game theme, you should know about spawn waves, difficulty ramp, and scoring systems. Basically, **be a well-rounded game developer** rather than only knowing one genre deeply.

- **UI/UX Design Sense:** Given the emphasis on clear UI and accessibility, brush up on **UI design principles**. Learn what color combinations are colorblind-friendly (there are public guidelines and tools to test contrasts). Practice choosing fonts that are easy to read on screen (avoid fancy illegible fonts, especially small). Understand how to layout a HUD so it's not cluttered. Also, be aware of **screen scaling and resolutions** (Unity's Canvas Scaler etc.) so that your UI looks correct on different screen sizes – the competition PC might have a different resolution than yours. You don't want UI elements off-screen or too small. These are often overlooked in prep, but can yield easy points.
- **MDA and Game Analysis:** Ensure you understand terms like **Mechanics, Dynamics, Aesthetics (MDA)** and other design frameworks so you can use them intelligently. For example, if asked or if you want to impress in presentation, you can say "Mechanically our game is X, which creates dynamic Y, leading to aesthetic (emotional experience) Z for the player." This demonstrates to judges that you have a formal grasp on design. Additionally, know how to identify **target audience** and design goals. Practice writing a one-paragraph "design brief" that includes target audience, genre, and core experience for random game ideas. This helps because in competition you can quickly formulate those points for your concept document and presentation.
- **Past Judging Criteria:** Memorize or internalize the typical criteria list (controls, mechanics, gameplay loop, camera, instructions, art style, clarity, audio mood, theme use, etc.). Use it as a checklist during development. It might help to actually have a printout of generic criteria (if allowed at your workspace). If that's not allowed, at least have your teammate cross-check verbally. For instance, before saying Module 2 is done, ask each other: *Does our game have instructions? Did we put in at least some sound? Is the UI clear? Can the player lose/win properly?* Proactively hitting each point ensures you're not blindsided in judging.
- **Time Management Strategy:** Plan a rough schedule for competition days and practice sticking to it in a mock scenario. For example:
- Day 1: 30min brainstorming, next 1.5h fleshing out design doc, last 1h starting project setup and maybe a basic prototype (carry to Day 2).
- Day 2: by mid-day have a minimal playable game (core loop working). After lunch, add secondary features and polish. Last 1-2h: fix bugs, integrate assets, add juice (particles, sound). Build and test before end of day.
- Day 3 morning: finalize presentation, maybe tweak small easy fixes in game if absolutely needed (but priority to presentation).
  Practicing a "game jam day" or two with these milestones can reveal how realistic your timing is. Also, learning to **prioritize tasks** is key: implement the riskiest or most essential features first (e.g. if your game needs A.I., get a basic A.I. working early; if the game's fun depends on level design, allocate time for that). If you run short on time, at least the crucial parts are done. Use the MoSCoW method (Must/Should/Could/Won't) before coding – list what absolutely must be in the game to be playable, what *should* be there to meet judging criteria, and what could be nice extras. This will guide you when time is almost up – you'll cut the "could" items and make sure the "musts" are solid.
- **Teamwork and Role Assignment:** Decide roles with your teammate ahead of time based on strengths. Typically, teams split into "programmer" and "designer/artist," but both should overlap a bit. For example, one focuses more on code architecture and mechanics, the other on level design, asset gathering, and UI implementation. However, both can write code when needed and both should contribute to the concept and presentation. Practice working together on small projects: do a weekend game jam together or build a mini-game in 8 hours to simulate the stress. This will highlight how you communicate under pressure. Maybe establish a rule like "constant communication": e.g. every 30 minutes, sync up for 5 minutes on progress and next steps to avoid diverging. In competition, stress can cause teams to go silent and just individually grind – that's dangerous if assumptions diverge. So plan to frequently check in ("Are you done with the enemy spawn? Cool, I'll hook that into the UI score."). Also, practice **delegating** – if one

is stuck on a bug for too long, maybe the other can take a look or you decide to bypass that feature to save time. Essentially, get comfortable working as a cohesive unit.

- **English Presentation Skills:** If English isn't your first language (likely in Finland, but Finnish competitors often speak English well), practice the presentation in English. Learn key game dev terms in English so you're not translating in your head on the fly. If possible, rehearse in front of others or record yourself to self-critique clarity. You want to be able to explain your game clearly and confidently. If you identify any pronunciation issues or habitual filler words ("umm, uh"), you can work on those. Also prepare a script or at least bullet points for the presentation structure we outlined and rehearse it with a timer. While you can't fully script the whole thing (because you'll be also referencing what's on screen or demoing), having a memorized opening and closing line can help nerves, and remembering the bullet list of points to cover ensures you don't forget, say, the theme or target audience slide.
- **Visual Design for Presentation:** Have a **PowerPoint/Google Slides template** that you can quickly apply that matches various game aesthetics. For example, maybe have a couple of slide themes prepared: a dark techy style, a playful colorful style, etc. or know how to quickly set up a master slide with your game's colors. Also, be ready to take nice screenshots of your game (know the shortcut keys, maybe stage some scenes for screenshots without UI if needed). If your game has animation or dynamic effects that are hard to convey in static slides, be ready to capture a short video or GIF. Knowing how to do that (using screen recording software, which might be provided or you can use open source one if allowed) is good – but at least a series of screenshots can illustrate different gameplay moments.
- **Documentation & Polish:** Small things count:
- *Comment your code* as you write, especially for any tricky parts or any borrowed logic. Not only does this help you if you come back to fix something, but if judges ever peek, they'll see you understood it. It aligns with the spirit of the rules.
- Keep a text file of credits for assets and maybe a short readme. Put this in the submission folder. It's not explicitly asked for, but it's professional and covers you in case a judge wonders where an asset came from.
- Organize your submission folder exactly as instructed – practice doing a final "export" where you gather build, project, docs, etc., in the right structure. A dry run of packing everything into a folder named correctly (e.g. "208_Pelituotanto_FI_FirstnameLastname_and_FirstnameLastname" as per 2022) will make it smoother at competition. Also, double-check that your game runs from that folder (path issues can occur if assets are not relative).

"Nice-to-Have" Knowledge vs "Must-Have" Knowledge

It's important to prioritize what to learn deeply versus what to just be aware of:
- **Must-Have:** - **Fluency in chosen Engine and Language (C# for Unity or Blueprint for UE).** You should be able to code common gameplay without needing to search docs constantly. This comes from practice – build several small games prior to Taitaja.
- **Gameplay Programming basics** (listed above: input, physics, collisions, UI, etc.). This is non-negotiable – every final requires programming these.
- **Team coordination** and source control basics. As mentioned, you both need to be able to work in parallel. If you've never used Git or Unity Collaborate, at least know how to manually combine projects (like exporting package from one project to import in another) if you split tasks that way.
- **Rapid prototyping mindset.** You must be comfortable making something that's not perfect but playable quickly. Perfectionism is the enemy in this competition. Practice cutting corners smartly: use primitives or placeholders at first, get the function working, then replace graphics or refine later. If you often find yourself stuck trying to get something "just right," train yourself to set it aside and move on, then revisit if time permits.
- **Stress management.** Competition days are intense. Must-have skills here include: taking brief

breathers to clear your head (even 1 minute eye rest), communicating kindly with your teammate even under stress, and keeping an eye on the time. If you tend to get very nervous presenting, practice grounding techniques or have notes to prompt you. Essentially, you must be able to function and make decisions while fatigued – simulate that in practices (e.g. do a long coding session then immediately present it to a friend to mimic Day 2 to Day 3 transition).

- **Nice-to-Have:**
- **Advanced Unity tricks:** ScriptableObjects for data, editor scripting to automate things, use of Unity's newer features (Shader Graph, Cinemachine, DOTS, etc.). These can speed up development or improve quality if you know them well, but they are not strictly required. For instance, Cinemachine is great for camera (and relatively easy), so that one is borderline must-have for polish. But something like DOTS/ECS for performance is overkill for a game jam scope – don't waste time on it unless you're already an expert and the game demands lots of entities.
- **3D Modeling or Digital Art skills:** If one of you can model in Blender or draw in Photoshop, that's fantastic – you can create custom assets (which can make your game stand out visually). But it's not strictly required since free assets are available. It's "nice-to-have" because a team with a skilled artist can produce some tailor-made graphics which might wow judges. Just remember to balance time – an artist teammate should still allocate time to import assets into engine and not spend all competition drawing one sprite. As nice-to-have, practice making **quick art**: e.g. can you sketch a character in 30 minutes that's decent? Or model and texture a simple object in an hour? If yes, that can give your game a unique flair. If not, rely on asset libraries and focus on integration and polish.
- **Pre-made Asset familiarity:** Knowing a few good free asset packs (like Kenney's 2D packs, Freesound.org for SFX, etc.) is helpful. Having them downloaded beforehand might not be allowed unless provided by organizers, but you can quickly get them if internet is allowed for asset search. Actually, in 2022 and beyond, internet use for grabbing assets was allowed, so plan some search keywords for assets. Nice-to-have skill: quickly searching and filtering assets (for example, typing "free medieval tileset Unity" or using Unity Asset Store offline documentation if available). If you know some go-to sources, you won't waste as much time hunting.
- **Presentation Flair:** Skills in PowerPoint animations or video editing to produce a short trailer are nice extras. If you can throw together a 30-second highlight reel of your game (using OBS or a screen capture, then trimming it), it could make your presentation very engaging. This is not required – plenty of winners just live-demo – but if you're comfortable doing it fast, it can impress. Similarly, minor showmanship like a brief intro skit or consistent humor in slides can make your presentation memorable (only if it suits your style – don't force it if not natural).
- **Knowledge of WorldSkills standards:** This is more meta, but nice-to-have is awareness of how world-level competitions evaluate game dev. It seems Taitaja is aligning to those standards (like quality vs quantity, emphasis on design rationale, etc.). Reading the WorldSkills Game Development test projects or forums could give insight into what is valued. Not mandatory, but if you know the international benchmark, you might prepare certain aspects (like accessibility, project documentation) more thoroughly than others might, giving you an edge.

Unity 2D Focus – Optimal Choice?: Yes, for most cases, focusing on Unity 2D is optimal. Unity 2D covers a huge range of possible game types that Taitaja tends to throw (platformers, top-down RPG, puzzle, card game, 2D physics games, etc.). By mastering Unity 2D, you automatically can handle UI, since it's the same canvas system. And you can even do "2.5D" (3D models on a 2D plane or vice versa) if needed. The development iteration in Unity is fast – play mode is seconds away – which is crucial for testing during the crunch. The community also has abundant resources (if you recall something but not exactly, the built-in Unity docs are available offline). Unreal Engine might shine if a pure 3D high-fidelity game is asked, but that's unlikely given time. Also, team sizes of 2 favor Unity's simplicity. If one member is an artist, Unity lets them drag-drop and adjust things easily without coding, which is very collaborative. So unless you have a specific reason to choose something else (and have practiced that extensively), Unity 2D provides the best balance of speed and capability. Focus your preparation there. Ensure you're comfortable with Unity's 2D physics (Rigidbodies, Colliders), tilemaps, Sprite animations, Canvas UI, and possibly the new Input System (or use the classic Input if easier for you, as long as it works). Unity 2D is definitely capable of making a product that meets all judging criteria, as proven by past teams. Of course, stay flexible – if 2026 surprise requires 3D (unlikely as that may be), Unity can handle 3D too, or you pivot to Unreal if you prepared that as a backup. But overall, betting on Unity 2D is a smart and "safe" strategy.

Final Checklist (Summary):

Before the competition, ensure you have checked off the following:

- **Practiced full-cycle game development** at least a few times (e.g. participate in a 48h game jam or simulate one). This should include planning, coding, asset integration, and presentation. Nothing beats rehearsal under time constraints.
- **Prepared code snippets/patterns** in your muscle memory (movement, shooting, spawning, UI update, etc.). You might even have them written in a personal notebook or cheatsheet in pseudo-code (if allowed), or at least ingrained in your mind.
- **Gathered asset resources** – know where to quickly find free assets (sprites, tilemaps, sounds, fonts) and know how to attribute them. If you have permission to bring asset collections (some competitions let you bring in a USB of generic assets that everyone could have), curate that carefully. If not, memorize some names so you can search fast.
- **Set up project templates** – e.g. a blank Unity project with common packages imported (if you're allowed to start from that on comp day – sometimes competitions provide a fresh environment only). Even if you can't bring a project, you can emulate it quickly (download Cinemachine, etc., if internet allowed, or have the packages on hand).
- **Delegated team roles and communication plan.** Decide who leads which module deliverable (one can take lead on design doc writing while the other proofreads and starts engine project, for example; one can lead speaking on specific slides while the other handles demo). Establish signals for asking for help or reassigning tasks if someone is stuck.
- **Studied criteria and judge mindset.** You should almost be able to quote the judging points – this way you won't overlook them. Also, mentally put yourself in judge's shoes: after your presentation, what do you think they might question or doubt? Make sure to address those proactively (e.g. if your game has a known flaw, acknowledge it and how you'd fix it with more time – better you say it than they note it first).

- **Developed speed and adaptability.** Maybe practice a bit of “code golf”: give yourself a small feature and see how quickly you can implement it. Or practice making the same mini-game in half the time it took before. Speed comes from familiarity and not overthinking – try to streamline your workflow (custom shortcuts, not obsessing over minor details too early).
- **Ensured personal well-being for the event.** Rest well before the competition, because you’ll be pushing your limits for three days. Plan snacks or quick meals if allowed (stay hydrated and keep energy up but avoid sugar crashes). Sometimes the simplest prep – like bringing a bottle of water and some protein bars – can keep your mind sharper than the team that skips lunch.
- **Attitude check:** Be prepared to be **creative, resourceful, and calm under pressure.** Competitions often throw curveballs (maybe software issues, or a theme you initially have no idea for). The prepared team has seen enough examples that they won’t panic – they’ll systematically brainstorm and come up with a plan. Go in with confidence from all your practice: you know you can do this. Also be ready to **learn and adapt on the fly** – if something new comes up (say, the theme demands a mechanic you never tried), you have the foundational knowledge to figure it out quickly (maybe use in-engine tutorials or docs if needed on the spot).

By focusing on the above, you’ll cover both the “must-haves” that will definitely be tested and “nice-to-haves” that give you an extra edge. Remember that success in Taitaja Pelituotanto is about being a **well-rounded game developer team**: code efficiently, design thoughtfully, art smartly, and present convincingly. With thorough preparation in all these areas, you’ll enter the 2026 competition ready to tackle whatever tasks come your way – and create a game prototype that stands out to the judges. Good luck!
