	XP Game Jam	hypothesis solo	Solo jam work
Hard skills			
Ideation/ brainstorming	- Pen and paper - Further designs explained / sketched on whiteboard - 1 team session, exchange concepts and pick one	- Pen and paper - Brainstorm action verbs / game genres / sketch concepts - Digital mind map with AI? - In theory open to returning to ideation, in practice expect 1 session	- Pen and paper - Brainstorm action verbs (mind maps) **- Describe game concept via Wh- questions** **- What to NOT do** **- Even if multiple concepts, what would make ME feel good about itself** **- Note MVP and minimal player flow steps** - Sketch game idea
Assets (VFX, SFX, UI)	- VFX/UI by team artists - SFX by team artists - Implementation/ positioning by me	- Assets by free sources (Kenney, OpenGameArt, game-icons…) + finding by me, time overhead - Implementation by me	- Assets by free sources + time overhead **- including script asset: CharacterController2D** - Implementation by me **- Style discrepancy or forced one-source assets even if unfitting**
Code prominence[1]	- Most code in C# scripts - UnityEvents and Inspector structs for animation-dependent actions	- Most code in C# scripts - Maximize non-code components and events for trivial tasks - Except for UI, do not trust UI defaults (LayoutGroups, EventSystem)	- Most code in C# scripts **- Main game loop with C# events** **- Not big enough to have substantial code or non-code parts**
Playtesting/QA	- Ideated as intermediate build test on day 2 - No successful testing before the deadline - No feature pivoting / bugfixing	- Individual concept testing via paper/mockup prototype? - No user/build testing - No feature pivoting / bugfixing	**- Early prototype – attempt for MVP after day 1** **- No testing: no individual concept testing, no second concept, no user testing** **- Slight upfront feature pivoting (against narrative dialog and multiple levels)**
Soft skills			
Communication	- Direct/immediate physical - Not always immediate (i.e. in the same room) - Partially segregated programmers-artists - Informal/unstructured	*N/A*	*N/A* - "Communication" with myself: scattered paper and .txt notes, Trello tasks
- Responsibility[2]	- The most experience = able to give advice, able to assert feasibility - Implement planned functionality - Implement given assets without bugs or artifacts - Presentation at final playtest	- Balance the workflow output between code/ functionality, visual appeal, design and user feedback - Achieve a complete, retriable game loop - Snap out of time sinks in mental blocks for ideation or for specific code problem	- ✓ Balance the workflow output between code/ functionality, visual appeal, design and user feedback - ✓ Achieve a complete, retriable game loop - Achieve a game progression extending past one screen and one level

- Accountability [2]	- No intermediate accountability (ala review -> if feature X can't be done, what action)	- No scheduled intermediate accountability	- No intermediate accountability (despite the Trello board)
Organization/ management	- No planning tool - No planning structures (stand-up, closure, review) - No dedicated manager/ producer/designer	- Planning tool – Trello? (after ideation / for development) - No planning structures - No dedicated manager/ producer/designer	- Planning tool – Trello (after ideation / for development) - No planning structures - No dedicated manager/ producer/designer
Adaptability	- Pivot towards MVP when the time is close to running out - No explicit process for prioritizing features	*expressed during project*	- Upfront prioritize and pivot towards MVP with help of Trello tasks **- Pitfall: too much time spent researching (physics) coding issues and adapting / cleaning up assets** - For coding, pivoted towards simple external asset
Circumstances			
Workforce	Team of 4 (2 artists + 2 programmers)	Solo (as programmer)	Solo (as programmer)
Environment	- Dedicated physical location (CMGT Ariënsplein) - Fluid schedule	- Digital (my room) - Fixed schedule	- Digital (my room) **- "Fluid" (scattered) schedule – 3x8 turned 2x10 turned 8+10+2 with sporadic breaks**
Workflow/ Tools	- Unity Engine - Git VCS + script collaboration - OneDrive (Web) / Discord files and todos - No planning tool	- Unity Engine - Git VCS - Notepad / Word todos - Trello planning?	- Unity Engine **- on day 2 was close to switching to Godot** - Git VCS - Notepad todos - Trello planning
Jam theme interpretation[3]	Literal (via fixed game loop with forced loss)	likely reinterpret / subvert	Literal with minimal subvert (via timed loop+randomized spawns with forced impossible 100%; subvert via **required non-100%**)

[1] Whether the game functionality was done entirely via code, or any substitutes were used (drag and drop, default engine components and events, visual scripting…). If any substitutes, assess whether they did the same work and whether their setup was faster, else assess whether the code could have been simpler and do the same work (relates to (over)engineering patterns and systems that might not be used in the jam game within the time limit).

[2] Responsibility and Accountability are treated as a subset of Communication concerning the internal and external expectations taken over in the professional role, and the moments (if any) where the state of reality compared to the expectations is reported for potential adjusting action.

[3] Given the theme, whether its literal or first impulsively obvious meaning guides the main gameplay or style in the created game, or figurative meanings or subversions are seeked.
(Example with "You can't save them all": a game with randomized spawning of objects that must be

picked up and led to a safe spot, with spawn rates balanced against the player abilities to always spawn more than the player can save, on a limited timer.)

Generally, for this benchmarking process, I extend "hard skills" to cover all steps of ideation-development-testing, regardless of their creativity or amount of technical(programming) work, and "soft skills" to aspects that concern work ethics.

Unanticipated differences between the hypothesis for solo work and the actual solo work are represented in the table **in bold**.

Benchmark 1: XP Game Jam vs hypothesis solo

The team setting at Saxion eventually expressed itself as typical, and from what I've heard and read, *stereotypical* for a game jam - gathering different skillsets on one (physical) place; sporadic, unusual work hours within the allowed time in the event; chaotic brainstorming process while getting used to each other; rapid decision making and feature cutting as the project needs to pivot in the middle. Between that setting and the solo setting at home, I expect many of the circumstances and the skills I will have to exhibit to differ.

Circumstances

I plan to use the same suite of software tools, namely Unity/Git and primitive tools for notes and file sharing, but I will have to adapt to the individual setting at home and I would approach the ideation process in a completely different way (at least, from what the team ended up executing in the first case).

As the table shows, I plan to follow a more structured work schedule; the goal is to work the same effective amount of hours (~24 like in the team setting), but simulate working days, which is unusual for game jams. Beyond that however, the setup is similar because I have not planned a time breakdown per game development phase; my only expectation is that out of 3 days, concepting might take the entire first day, in order to choose an idea I confidently want to work on, and to sketch its gameplay specifics. I already suspect this might be necessary because a single session of brainstorming with a time limit might not result in a game concept I want to work on, even if it forces me to pick one and “save time” by moving away from concepting early. The fact that an inspiring concept might come outside of scheduled work or creative phase is also explored within the following article, for example “It wasn't unusual to blow the first 3 or 4 days of the week just fooling around watching O-Zone music videos…” (https://www.gamedeveloper.com/game-platforms/how-to-prototype-a-game-in-under-7-days).

Hard skills

For ideation, I would approach it similarly to my *initial* approach to the team brainstorming before the team format took over: structure my first impressions from the jam theme formulation under one or more mind maps that interpret the phrase (here: "You can't save them all") or its words as broadly and varied as possible. What can "save" mean? Who can be "all"? I expect a game concept to form out of that especially because I believe this process will more easily find ways to subvert the immediately obvious interpretations of the theme, which I enjoy seeing and developing in games. For comparison, the team concept ended up being similar to "Papers, Please", in which the player is

faced to make choice after choice between morally good and legally correct by the rules (in our case this is even more gruesome since either choice would have led to a quick reveal that no one is being saved). Other too intuitive concepts I saw among the final jam games included: timed games with randomized spawns to lead to a target location with too many spawns for the given time, and forced usage of companion units to beat difficult levels and then be scolded for not protecting (killing) companions.

Regarding visual and audio assets, I know I am unable to create them myself within the jam time. I will likely have to use primitives for prototyping, for which simple drawings or geometric shapes and no sound *are* acceptable; however, then I have the choice to not advance the audiovisuals further or hunt free assets. I will prefer the latter, even though I expect it to take me an extra task's worth of time, because I still want to create a finished game (as much as possible) during this experiment and not simply a functional prototype. In this sense, just receiving assets from artists in the team setting is much more convenient, but I still have to implement them in the game regardless of setting. In fact, I expect a small benefit from the fact that I could cherry-pick assets of specific sizes and (likely) not have to handle odd image sizes or file formats. In the team setting, this is a matter of communication in the best case and rushed importing in the worst case, either way taking more time.

Coding is under-addressed in this benchmarking for several reasons: it is my core "hard skill" and I expect to gain more new experience and reflection insights on any other skill; it is hard to describe the coding process upfront in a sensible way (for more rationale see the part below on game design); usual "objective" descriptors such as "code quality" are not the target here, because a game jam is an event of fast decisions effective in the moment. I do try to comment on one aspect I expect to optimize, which is minimizing direct IDE code through any applicable tricks like drag-and-drop or Inspector events and settings, and relates to preventing overengineering or excessive boilerplates for simple functionality. In the team setting, I set up such systems myself with the intent to be designer-friendly, and ended up using them myself, for which I was glad to not have to tinker values directly in code, but in the solo attempt I hope to go even further by (ab)using Unity default functionality as much as possible instead of coding it anew.

Playtesting was criminally underlooked in the team attempt due to the haste of the game jam and the overscoped idea, and I have not strictly planned it within my solo attempt either. Part of the reasons is similar (if an idea falls behind schedule, time should be saved first from the non-MVP parts), and part different (in my "home" setting, it is much more difficult to find able and willing people even if everything would be smooth on my end). Any potential "testing" would be my own testing on whether I like the prototyped concept, in minimal possible form. Non-critical bugs will not be fixed, with the assumption that there will be enough critical bugs or unfinished features for this to be a minor point.

Soft skills

Logically, in the individual setting there is little to talk about *external* communication, expectations or even organization. Instead, "communication with myself" materializes for me in the notes and considerations I leave for myself. I would rather consider this a non-immediate form and label it inferior to the direct, explicit communication that naturally happens with other people physically in the same place. However, the team setting showed me that this skill also needs to be practiced, since insufficient (not enough, not professional) communication between us furthered the initial overscoping problem. (But logically, I do not expect I can practice this skill alone.)

Interesting is that responsibility, accountability and organization end up with similar descriptions. I do not expect much planning, let alone conveyance of progress in an individual setting. I should however look out for two things: first, in the individual setting this need not mean “never stop to look back and be aware of your state of progress“ – I just expect that I should develop a skill for having these moments at the right time without formal ceremonies. Second, in the team setting this should not have been the case and some moments of accountability should have been formalized. In the absence of a dedicated person to manage that, someone present should be mindful of that, and I have the awareness but did not express the necessary actions. This also points at my perception of responsibility regarding taking on important missing roles – in the team case, as the one with the most experience this time, and in the individual case by minding all tasks upfront for a balanced product with a finished feel (sadly for my hard skills, a game isn’t all coding either!). The very simple takeaway here is to raise my voice more – whether to teammates, or to the part of myself that might be pushing through constant development under crunch.

Summary

In summary, I expect my solo game jam experience to differ from the on-location experience in several ways: ideation process, obtaining assets, communication, responsibility, work schedule. However, some aspects will probably be similar: asset *implementation*, (lack of) playtesting, organization/management, as well as the used tools (by choice). I anticipate that there will be a tradeoff between team and solo settings, but not in an intuitive “linear” way – for example, the accountability to myself being no less than the one achieved in the team, or assuming I will plan with *more* structure than in the team attempt. This shows that game jam attempts are unique just like business project cases, and training for successful communication and calibration between multiple people is important.

A word on game design as a comparison aspect

Game design is a major part of the rapid development process during a game jam - poor, overscoped or underspecified design may lead to technical tasks spiraling out of budget or unimplemented audiovisual assets. However, it is not considered or planned for here because I expect the design process and any specific decisions to happen only in real time, emerging only from the concrete game idea and possibly time restrictions. Unlike criteria such as Communication, I believe this to apply for *all* cases that could be entered in the table. Whether this is a weakness of mine as a non-designer or an inherent property of game design itself, that is, whether the process of designing can be modeled upfront without a concrete game idea, is up for debate.

On a similar note, listing "Jam theme interpretation" describes purely the preferred approach to jam themes of the person with the idea being developed, but does not describe skills in implementation, stylization or communication, and so using it as an upfront criterion to imply a good plan to follow makes little sense.

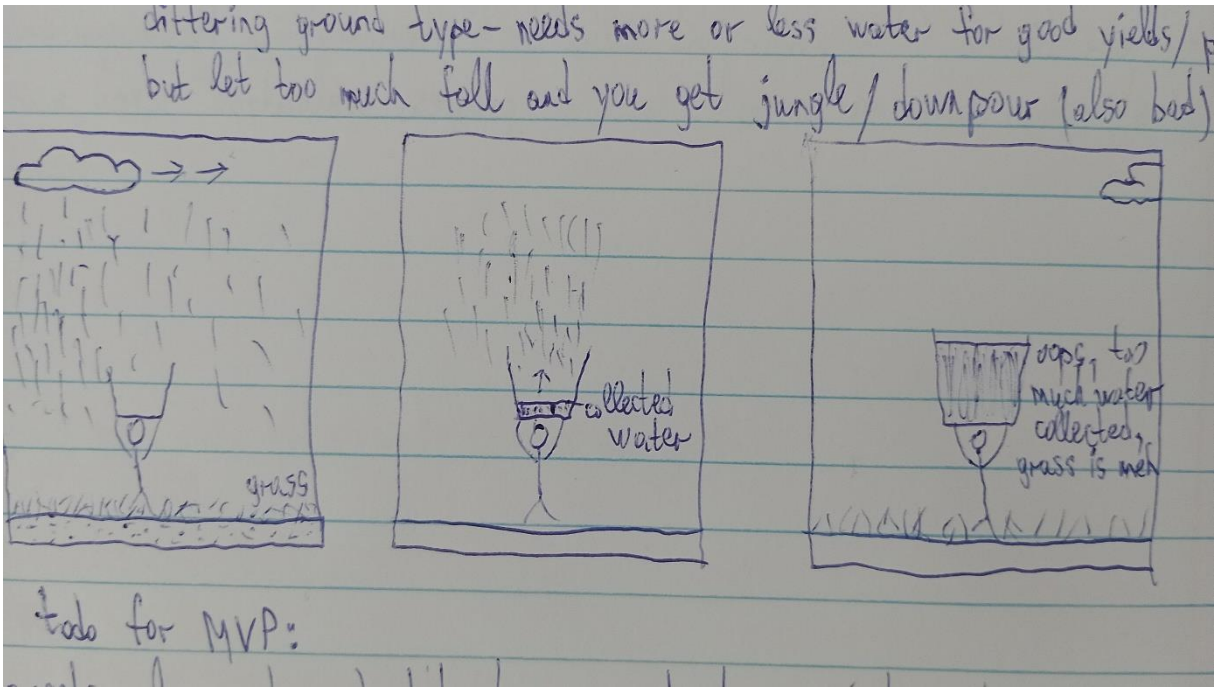
Benchmark 2: hypothesis solo vs real solo work

Circumstances

My initial idea for holding a work schedule of 3 days x 8 working hours fell apart surprisingly quickly with the start of development. One reason for that was the workload from other tasks before starting the game closing in on the available time, which made me imagine I would do a 2x10 instead.

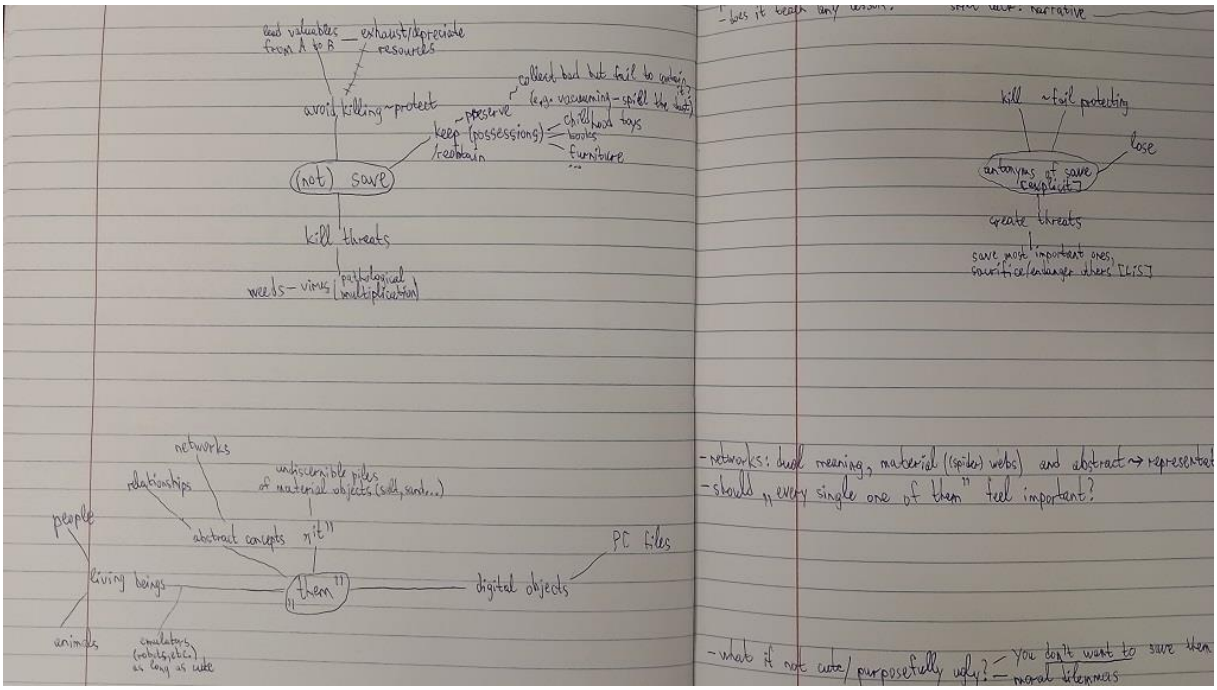
The final 8+10+2 hours ended unintendedly close to the typical game jam format of “no schedule”, with similar perceived crunch pressure. The other reason is related and is about breaks – less than one hour after I started the ideation process, I felt the need to take a break from the screen altogether, and it was during this wind-down in bed that I came up with the idea I developed later. This all sounds quite similar to the organic game jam experience and the one linked earlier in the article (https://www.gamedeveloper.com/game-platforms/how-to-prototype-a-game-in-under-7-days) – in other words, sometimes breaks *are* “work”, and not all work hours weigh the same. I committed to this with further unscheduled breaks, so my experience here ended up the same as the team one.

My resulting interpretation of the theme came to have little subversion – timed rounds with randomized spawns that ensure “you can’t save them all”, but my twist was “you *don’t need to* save them all anyway”. As shown below, the player collects raindrops for drinking water, but must also leave a sufficient percentage for the ground itself (here: grass) to remain healthy.



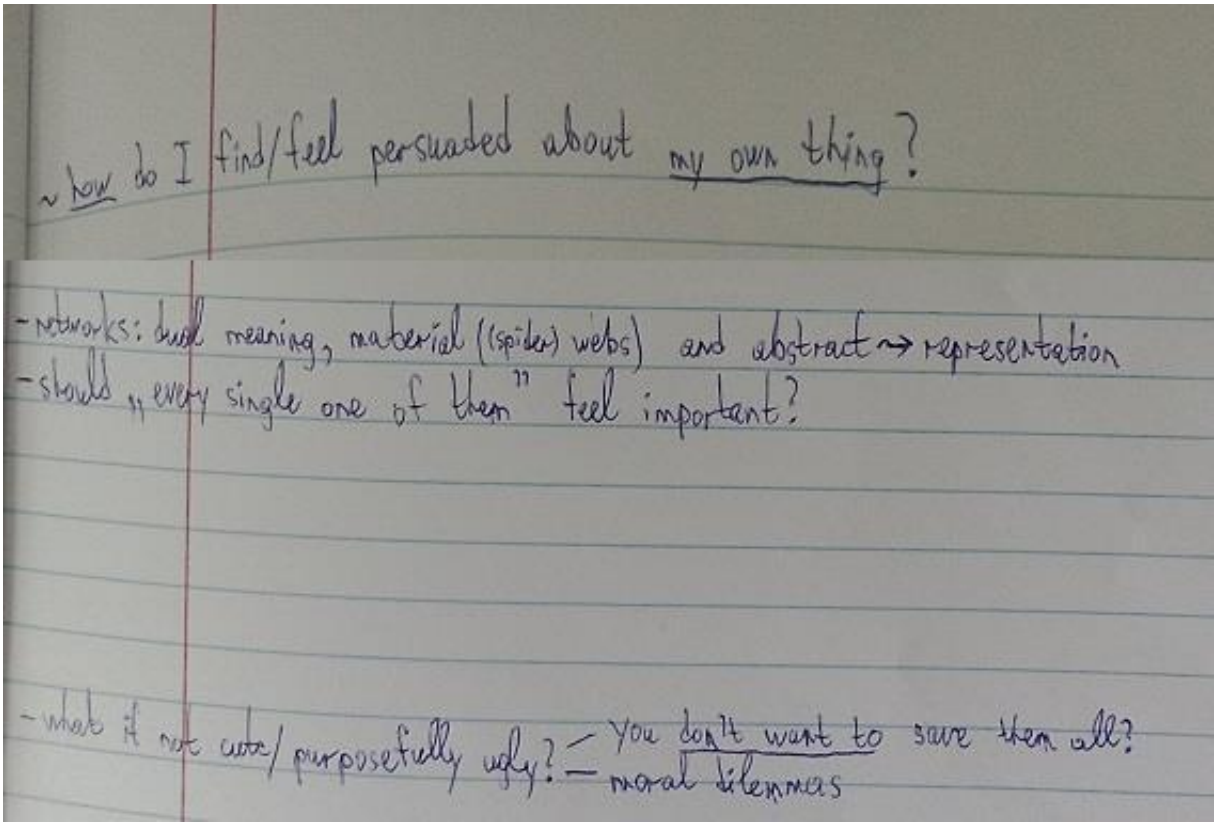
Hard and soft skills

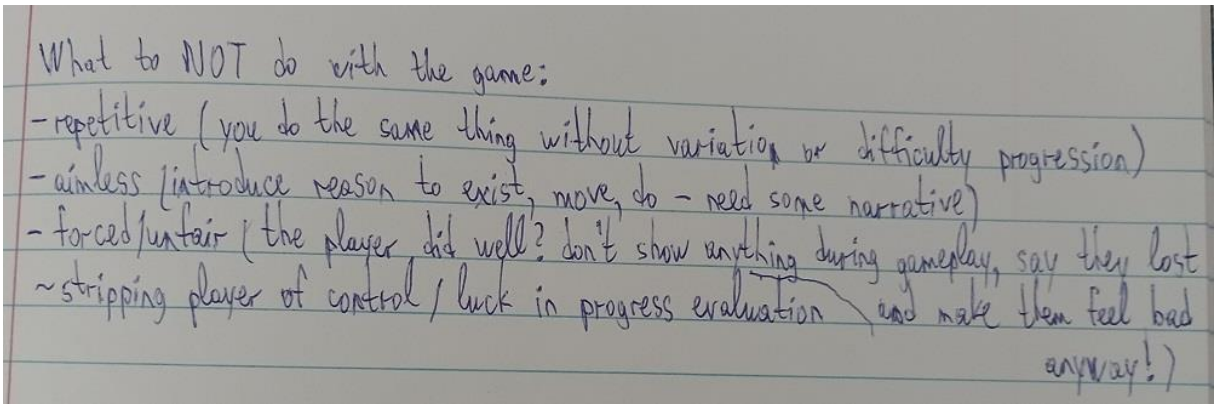
The ideation started as I expected, on my own terms – encompass the theme fully, make sure I “understand” it (in this case, interpret it as broadly as possible), and try to work from there on with the prompts (my upfront planning stopped at this point).

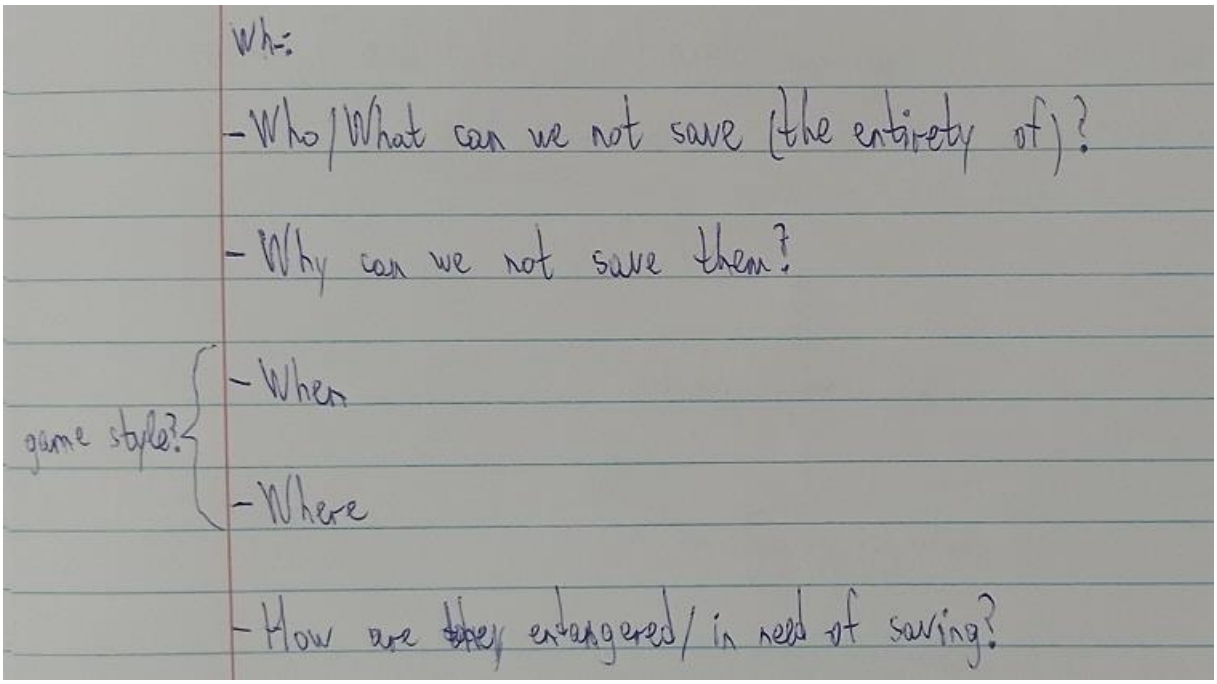


Interestingly, the actual process also stopped shortly with a limited number of mind map items – at this point I felt I was not coming up with any other quick ideas and took the already described break. In result, there was no multifaceted ideation work, no multiple concrete ideas to pick from or a fallback in case a prototype failed, but the one idea did the job.

A few more unplanned thought sidetracks also happened, centered on what *I* want and don't want out of the game, as well as Wh-questions as a tool to put the abstract concepts in a context (and possibly help narrative or style later, if I would implement any):



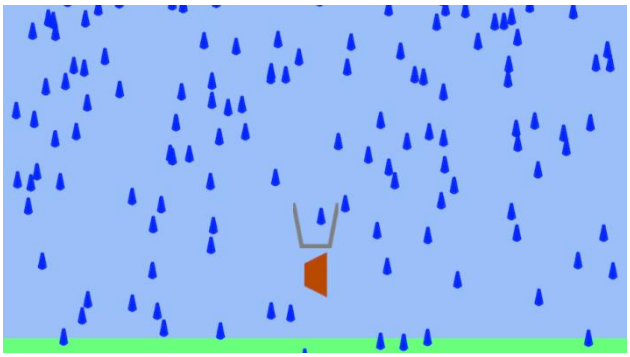


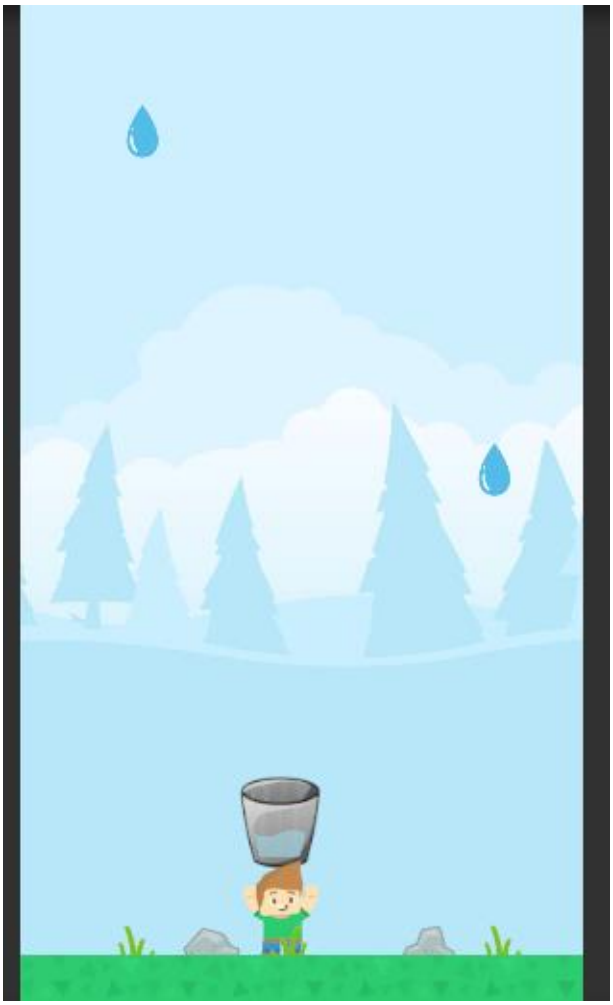


In assets and code I encountered two major setbacks. First, the game idea was simple enough, scoped, but I still had to implement player movement – physics? Unity sent me down a rabbit hole here with its requirement for Colliders (and Rigidbodies) to move and actually collide (in short, usually only a dynamic Rigidbody moving into something else will collide, and I want to fully control my player as a kinematic Rigidbody). I faced my first pivot decision – should I start anew in Godot Engine, where I just learned the basics of physics collisions and could better deal with this? Ultimately I decided against it as an overkill, and instead adapted a simple external CharacterController2D asset to "fix" this.

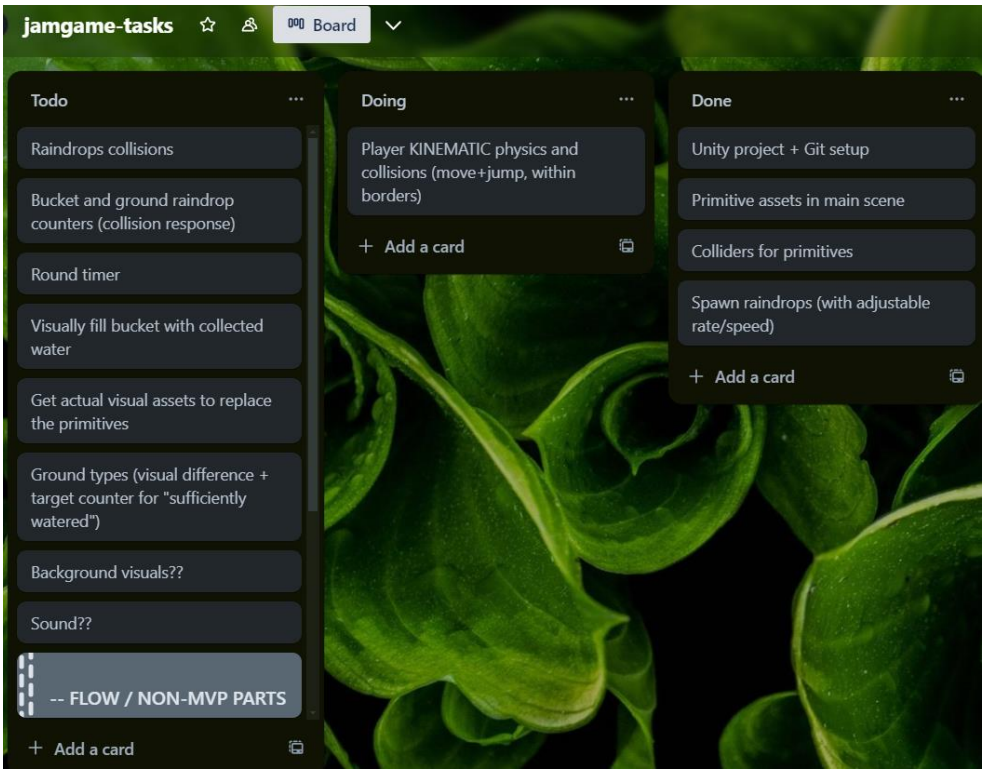
Second, when transitioning between self-drawn prototype and final assets, I could not represent all assets I wanted to have with only one package (at least from those I found). This meant a mostly forced style for the game (most final assets are Kenney assets) and the missing ones I needed were forced to not deviate too much – so I had to be careful with the bucket and raindrop assets, for example pixel art was out of the question by then. To stick with the "dominant" style defined by most of the used assets so far was a design choice with which I ensured not spending time I didn't have for asset hunting, and I believe the result is *passable*.

Other than these setbacks, the coding actually remained so simple that Inspector references and C# events were already the hardest part of setting up, and so there are no notable extra drag-and-drops or events to talk about, despite my expectations.





One crucial positive difference in the solo project was the task planning. After the ideation and first prototype work, I was in a tired and reluctant state, but still I made a Trello board. The tasks are minimal, I cut all corners – all “descriptions” were the task titles, no comments or checklists, no user stories or exact specifications of “done”. The minimal grouping I did was to plan a section for non-crucial parts and parts that *were* necessary for the game flow, but not for finishing something *minimally playable*. Even this much proved to be invaluable in the remaining time, as I was capable to think of the programming and scene UI solutions of each task, but lost awareness of the specific tasks remaining almost always after looking away from the Trello board.



This boosted my self-management and adaptability. Even though I was successfully decently in control of the scope from the start with my small game idea, I had little time, and I explicitly re-prioritized a few hours before the end: level UI was promoted to a necessity to finish, ground types and narrative demoted to optional parts.

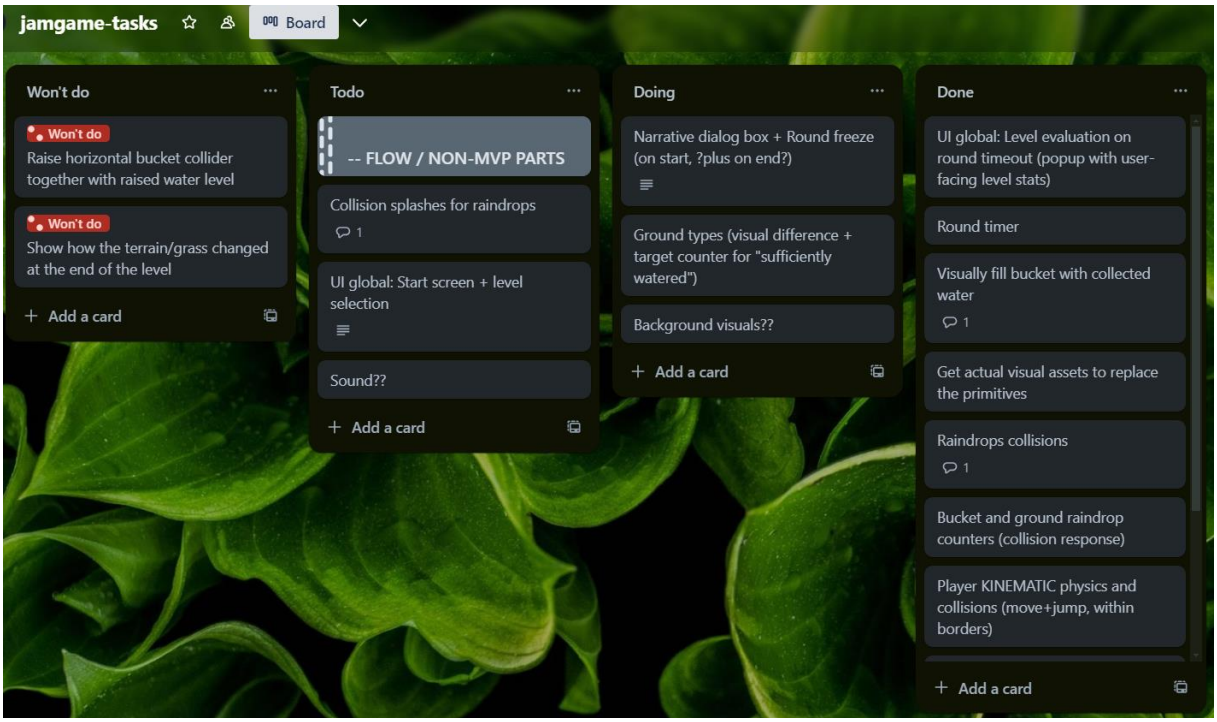
Even with planning however, I was not 100% time-effective. For example, I spent extra time considering assets to import granularly or explicitly deleting “unused” assets even during the last 2 hours I had. I notice this can be categorized as *discipline*, for which responsibility and accountability can help, but do not fully cover, and need to be very strong at a team level to suppress my own long-standing urges during a project (e.g. to write scalable but complicated code, to try optimizing, to delete what I don’t use…)

Summary

In summary, I am satisfied with the resulting game from my solo game jam experience. I attribute this to good adherence to solid initial plans and values (with which the hypothesis in this very document helped, which is unusual for a typical game jam!) as well as good decisions to deviate from the plan when the reality necessitates it. I can reinterpret this as “good organization, communication with myself and good adaptability, even though I still went off track at several moments and could minimize that”. One important takeaway that is impossible to re-apply to my solo setting is to *voice myself in a team setting with my concerns and ideas to organize and pivot as much as I did to myself this time.*

Even the planned tasks for this game were not all finished at timeout. I already had concrete ideas for how this game can be expanded in content and improved in feel and fidelity, as shown on the final Trello board below. A friend outside of the study also showed interest in the idea with further extension ideas. This puts the game at an interesting crossroads – I would feel glad to continue it beyond the “game jam”, but first, I would compensate for the missing playtesting. I could say the

game is almost at a “vertical slice” state – if players would not like to engage for a minute in the concept in the one level, there is little sense to make the originally planned variety for multiple levels, the corresponding UI, or sounds and particle effects, so in the current position I can cut my losses early.



Improvements – summary

*shortcomings **illustrated in the resulting games** turned into action points.*

To summarize my lacking skills as found by reflecting on my game jam experiences and what I could do next to improve on them:

- Implementing fundamental features in games quickly, without problems and sidetracks – do more small projects with free external simple script assets in order to get used to the thought that using them is a valid option, OR do more small projects until I develop a library of self-made fundamental scripts;
- Avoiding style discrepancy common with found free assets – within game jams or similar experiences, proactively contact artists or write an open post, AND explicitly talk about the styles I would be most comfortable working with (before and during development);
- Ensuring playtests – individually, spend even less time on unimportant details during an MVP phase (example from this time: how to draw 2D polygons in Unity -> just use a simpler shape); on a team level, enforce a planned intermediate moment for playtesting with other people, set it in stone in writing, Trello, or similar;
- Avoiding communication shortcomings in a team setting (unstructured with things "lost over the wire" across team members, rooms, or phrasing) – talk more; do not give up on asking for clarifications if something is unclear; if it frustrates others that they have to explain it too many times, nudge towards using a different approach; for any executive decision taken with less than the full team, write down for the others+contact them immediately to discuss and point at the writing; schedule moments for progress accountability in front of the others before the start of ideation;
- Recovering speed from getting stuck in tough coding problems or distractions (fast adaptability) – write down (assuming more time allotted) tasks with concrete minimal conditions of "done" so that I can refer back to what I should actually achieve (solving the XY problem) + develop my brain process to stop and look back at what I am trying to achieve.

As an even shorter summary, the biggest takeaways are "gain more experience – best done through *short* games" and "communicate explicitly, always – until I become effective at it".