POST MORTEM - GAME PROJECT 1

Project Title
Akined

Date

18 October – 12 November 2021

Project Members: Oskar Nilsson (PM)
Vincent Nord (PM)
Wes Ekman (GAUX)
Joakim Hedman (GAUX)

Project Overview

Akined is a 3D Isometric Puzzle Platformer where the player takes control of the old man Adam and his Spirit to solve puzzles using the power of Possession, to control objects in the environment.

We set out to make a compact and clear puzzle game using almost exclusively 3rd party assets for faster iteration and development with goal to always show the entire play area from the start and through that use the isometric view to our advantage.

The target audience we went for was 'Teen', but we of course wanted the game to be playable by all age groups, so no strong language, gore, or adult themes should be present, though we wanted a bit somber and sad story element.

The core game design pillars were:

- Challenging, not punishing: All levels should be completable by all skill levels, though the puzzles should not be too obvious.
- Somber, but whimsy: Somewhat dark and brooding aesthetics but with fun and fantastical elements.

Project Details

Schedule

We began our initial planning and schedule to have idea brainstorming, set up project specific details (Git, File standards, Code standards, etc), and team-building exercises for our **first week**.

The roles for each member came about from discussing previous experiences and personal likes/goals. My role became somewhat of a *technical designer*, mostly because I had the most programming experience, though I cooperated with our only programmer to discuss and set up coding/file standards and Git repository.

Our combined goals were to have:

- No crunch/overtime, but flexible time management, so if you had used an hour of the working day for something else then you would try to make it up in the evening or another day.
- Daily stand-ups where we talk about what we did the previous day/week, what we were going to do current day, and for some minor team-building talk; what we did last weekend or what we ate for breakfast.
- Transparent communication where everyone could talk about what they wanted with our project and trying to discuss everything as a group to reach a goal together.

The **second week** would then be used for actual pre-production and going into actual production with the goal to have a playable prototype by Friday.

Our **third week** would be fully in production phase, while the **fourth week** would be focused on adding last crucial elements (if needed), fixing bugs, polishing, and preparing marketing materials and a final build of the game.

Polishing could also be added earlier into the production if we felt that the main game loop was finished enough.

Our thoughts at the beginning were that having a bit bigger planning phase would be to the detriment of our project and production since most of our group didn't have any experience in making games, but we discussed it and set out to have a decently planned idea and realistic goals by having an entire week for planning and ideation, which would make things a bit more clear on what features we actually needed and what needed priority.

Development & Project Tools

For this project it was required to use **Unity** and recommended to use a LTS version of *Universal Render Pipeline (URP*).

We diverged at bit from the recommendation and used the *Built-in Render Pipeline* with a LTS version **2020.3.20f1**. This decision was mainly based on my own recommendation to the group since our scope was quite narrow and I felt that more graphical fidelity of using *Shadergraph* or additional Post Processing additions wasn't necessarily needed for what we intended to do. We also talked about using real-time lights for better mood lighting, and from my brief research it looked like that URP didn't support real-time lights anymore.

The **project management tools** we used were:

- Trello
- Miro
- Google Drive documents
- Gitlab (Version control)

Technical tools used:

- Visual Studio Code (IDE)
- SourceTree (Git)
- Adobe Premier

- TwistedWave.com (Online audio editor)

Unity packages used:

- Post Processing stack
- Unity Recorder
- ProBuilder
- TextMeshPro

Asset packages used:

- POLYGON Dungeons (*Synty Studios*)
- POLYGON Pirates (*Synty Studios*)
- POLYGON Particle FX (*Synty Studios*)
- Basic Motions (*Kevin Iglesias*)
- Pose Editor (*Sator Imaging*)
- Text Animator (*Febucci Creations*)
- Quick Outline (*Chris Nolet*)

Other assets:

- Various music & sounds (Freesound.org)
- Narration/VA (Joel - GAUX teacher at FutureGames)

System Requirements

We didn't do any thorough testing of hardware or software, but I developed and played our game on my 8-year-old gaming laptop (*ASUS G46VW ROG*), and it ran decent but with some stuttering, having the system specifications:

- Resolution: 1366x768
- CPU: Intel Core i7-3630QM
- GPU: Nvidia GeForce GTX 660M, 2GB DDR5
- RAM: 16GB DDR3 1,600MHz
- DirectX version: DirectX 12

We only made a build for the Windows OS, and the only included way to play the game currently is using **Keyboard** and **Mouse:**

- Keyboard: **WASD**-keys, **Q**-key, **E**-key, **F**-key, and **Spacebar**
- Mouse: **Left Mouse button**

No internet connection is required to play the game.

Known Bugs

- Animations between Adam and Spirit must play out and becomes somewhat queued if spammed.

- Possible to change to Spirit at ending scene (Doesn't really have an impact and doesn't break the game but it is pointless as there is nothing to possess in the scene).

- Possible to pause at transition states between restarts/scenes.

- Bridge-extending sound played every time Valve is interacted with, even if there is no bridge in current scene.

- If spam-clicking the 'continue' button in the dialogue frame the VA (voice acting) will be overlapped (Due to some sounds taking longer time to being read-out than the text in the frame being written-out).

Project Outcome

Schedule

The **first week** went as planned where the first 3 days we brainstormed and came up with an idea and got the clear from our teachers to continue and work on it, and the rest days of the week were focused on planning our timeframe and deadlines for the first prototype, playtests, feature- and polish lists – that we will get to later. On Thursday or Friday, we also did set up a project folder that was version controlled to our Gitlab repository and setup our general folder structure.

Our initial planning did work out as we had stand-ups every day, however with having one person not reachable at all and another not on location proved to be hard both for the reason of our plans being a bit up in the air and hard to gauge how someone was doing on tasks.

Going into our **second week** we began a bit of pre-production according to schedule, and we managed to divide work into chucks of what we felt was fair, however some question marks about how and what to have the Project Managers (*PMs*) handle, especially when we would reach the point in production where implementation would be more important than the management of it. These questions would be put on hold for us to handle when we got to them later.

At this point we had a playable prototype, but not to the degree of letting play testers yet have a go at it.

In the **third week** some problems occurred, such as our second, and only programmer left/quit the education. This resulted in that we had to reschedule things a bit and that one of the designers with the most programming experience had to focus on the bulk of the programming related features, and some features had to be put on hold such as sounds and VFX.
Regarding the schedule, we basically restarted most of the tasks on Trello and the entire group talked about on what features to put on ice or cut, as well how minor programming tasks had to be left for our less programming-savvy project members to handle because of how much our now new lead programmer would have to do.

In the end this resulted in that the main tasks of game implementation were divided up between the two designers, while the other two PMs had to focus solely on asset acquisition, marketing materials, and presentation for the last week. The polish and bug fixing were pushed to the 4th and last week instead.

At the end of this week, we also had some play testing with people from other classes at the school, such as artist-, UX-, and project manager students and teachers, and which gave us some good feedback and a few wake-up calls/critical faults in our design that we would have to think about.

Last week of the project became a bit jumbled where basically all tasks on Trello were active because all project members had to jump between each task to discuss and organically solve things as a group, be it making the poster or trailer, to the implementation and feel of certain gameplay features. Regarding how successful or close we were to how our initial schedule was set up, then it could be considered somewhat successful in the worst ways since our worst case ended up being the actual reality where there was a struggle to get things working and polished.

The final build, with most of the features implemented and crucial bugs fixed, was made within an hour of the required delivery date/deadline.

Technical Issues

The majority of technical issues came from the group's inexperience with Git and SourceTree, even though a few members had previous experience in using it though mostly on solo projects. The issues ranged from not being able to clone projects, to basic but somewhat strange merge conflicts where changes were missing.

Some of these problems were solved and handled through talking things through with each other and finding ways to circumvent them happening again by separating content better in branches by committing each completed feature instead of all changes inside one commit.

Other issues came from one of our PMs had a Mac laptop that caused some issues that new changes inside the repository didn't show up for him specifically, but this was simply and annoyingly solved by having him clone the same project over and over to get he newest changes. Luckily the PMs didn't need constant access to the project since they didn't do any implementation of features.

Design Changes

One of our biggest changes to our design was that initially the old man Adam and the Spirit was supposed to have a tether between them and linking them, and if they got too far from each other they would die and would have to restart. This had to be removed because of both time constraints where we had to push this feature back more and more, and because of technical reasons when no one in the group had any good idea how the visuals or the implementation would work, especially in such a short time.

Early on in our production we changed from a perspective viewpoint to isometric instead, our reasoning being that things would look a bit more stylized and feel like things had better intents using it. We also felt that it would be appropriate to our scope as well we could use it to our advantage by hiding or emphasizing objects or areas in the environment.

What Went Wrong

By far the biggest hurdle to overcome were both of our programmers leaving/quitting and our group having to replan, redesign, and reschedule a lot. Thinking back, we probably could and should have had more focus on interacting with people who weren't on location, as how we did things was that we had our stand-ups at the start of each day, and we asked everyone if they knew what and how to do their tasks, and then leaving them to their own devices. This is where we should have been more obstinate about trying work together and seeing each other's work at the end of each day, or at the very least having a lot better communication in general.

What Went Right

Our initial plans by having a good small-ish scope with basically only one main mechanic proved to be a big blessing for us. For the entire first week we designed, planned, and scheduled our production phase, as well as trying to think about playtesting, feature locks, and final build, as is something that should be taken to heart continuing forward.

Additionally, after our struggle with having to work without programmers something good came from this – our group began bonding even harder and became a lot more focused on finishing critical features and being honest with each other.

My personal reasoning behind why our group's mental state change so quickly was probably because of now having all active members on location we felt that we could talk and have fun with each other more easily and we had now reached a low point, so we needed to be pragmatic about our development choices.

Conclusions

Summarily, we managed to create a decently fun and interesting game which ended up winning a few awards at the school presentation. An important take about this is that none of our group members expected to win because of our early problems, but of course we had high hopes and felt proud about what we had made.

We had to make quick decisions both technical and design-wise for our game to be consistent and cooperation between all skill sets where everyone had to do something they weren't comfortable in doing, or had no skills or experience in, and this was most likely the biggest learning moment for all of us – having to learn and do things you've never done before, and also because we had a deadline so things had to be done in a timely manner as well, so a true creative approach came about from this.

Some points to consider for the future for ourselves or others going into development:

- Set up a concrete plan-of-action of your most important features and elements; try to leave nothing to chance and set your scope appropriately to you or your groups skill levels, so if something unsuspected happens then you should still be able to solve or complete your project in a timely manner.

- Put heavy emphasize on connecting with each other, especially if it is a divided group with people at different locations. Try to show your progress and encourage each other the entire way while still being transparent and goal oriented in your feedback.

- Kill early and kill quickly; don't let unrelated ideas or extra features be implemented unless it can be play tested early or can be proved in some way to improve your end-product. Remove features that gives nothing except bloat for your development team, even if those are dear or darling ideas.

- Do not underestimate paper prototyping and play testing because it can, and most likely will be a great source of information and perspectives that you hadn't thought about. If possible then do it early and find those edge-cases or special tidbits that could be improved upon.

My personal thoughts are in the end that projects with deadlines are the best way to learn fast and improve creatively how handle problems, both in your day-to-day life and professional. It's a great way to grow and find new perspectives both from other people and by self-reflection.