The Florida Historic Capitol Museum

Apollo 11

Interactive Exhibit

Write-up

Arcvale

Summer 2019

The Florida Historic Capitol Museum

Introduction

This is an attempt at retelling the journey a small team of game developers took to attempt and create a unique experience for a museum with a short deadline, no budget, and questionable time commitments. I have done my best as director to attmept and catalog all of the events that took place during this project, and have attempted to give credit to all the responsible parties for their glorious and painstaking work in making this project a reality.

It is possible I may have missed tasks that were completed, or additional work that was done by one or more indiivudals. I apologize if that is the case. This write up was done in tandem with going through all of the team logs, meeting logs, and the entire chat history that the team put forth. If by chance I missed something imparative, I apologize. This post mortem was written after the fact during the weeks following the inital deployment, as there was little time to take cohesive notes during the actual creation process.

The Request

It began with a request from the Florida Historic Capitol Museum. The 50th anniversary of the Apollo 11 landing was coming up on July 20th, 2019. In honor of this event, the museum was planning on creating an interactive display on the topic as their next rotating exhibit. They already had several interactive examples, plenty of informational material, as well as some technology-focused creations they would be exhibiting.

Two weeks before this deadline, I was asked to come in and meet with them about creating an idea they had for a sound board revolving around the moon lander that one of the curators had seen online, and wanted to replicate. (Fig A1)

Fig A: A screenshot from the video the curator had seen online of the device she wanted to re-create.

The beginning

The initial meeting with the museum established two very important things, one being that the deadline was absurd and in literally two weeks. And two, the museum had no concrete plans on what they wanted for our section of the exhibit, so we had complete creative control.

This is arguably the worst possible combination that can happen in the world of game design, as the only limit is time. No console limitations, no target market aside form ‘people who go to museum and their kids’, no budget, no nothing. Just ‘get it done as fast as possible’.

So of course the reasonable response is to bite off more than can be chewed and the entire team loses both sleep and insanity as we choke on what was supposed to be a simple deployment.

Assembling the team

I had only a few days to get an entire cast and crew together for the project, which made things much more challenging. It’s a bit of a balance between two equally unappealing outcomes. Spend too little time getting the right crew, and the project falls apart. Too much time, and there isn’t enough time left to complete the deployment.

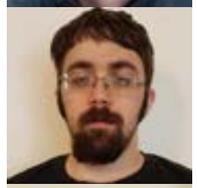
This of course was made harder if only because of the lack of budget. Anyone working on this project would be doing so for free, if not at the expense of their own time and materials with almost 0 chance of reimbursement. Plus the deployment would only be for the museum’s Apollo 11 exhibit, which lasts only 5 months before getting taken down. So it wasn’t exactly easy sell.

Luckily I was able to get in touch with some of the individuals I’d worked with previously on other game deployments and convince them that this wasn’t the crap shoot it so obviously was.

I was able to get the following people from past projects:

Keith Roberson

The legend himself. A 20+ year proffessor at Florida State University who teaches for the art department. He’s been doing art exhibits like this involving tech for decades and was, luckily for us, looking for a short project to hop in on.



Alex wood

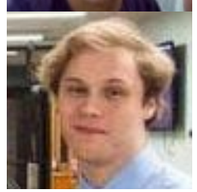
A longtime associate of mine who specializes in animation. He’s lead the 3D animation teams on a few past projects I’ve been on and does 3D modeling when needed. He also has a nack for programming, but prefers to stay on the art side of things.

Christopher Tonner

The sound designer and foley artist in Nashville that runs a studio doing this kind of thing. He’s a one man powerhouse, and makes all his own sound effects from scratch. We joke that he has a fetish for high frequency sine waves.

Ethan Shelton

One of the better programmers I’ve ever worked with. He’s one of those guys that forgoes sleep to learn some new piece of code that optimizes a program by 0.04%, and has been the saving grace in many past projects.



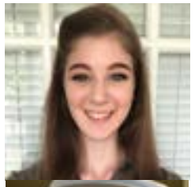
Bryan Clark

An associate of mine from way back who specialized in 3D art and design. He’d been on a few projects with me in the past, but had been busy trying to get a double degree for his undergrad in two less-than-forgiving subjects and hadn’t had a chance to work on any actual games in a while, so he was eager to step in for this one.

However, getting these people to work for free was about all I was capable of. Anyone else I reached out to from my past game deployments was less than thrilled at the lack of budget or direction, and even less so at the very rapidly approaching deadline. I didn’t blame them.

Luckily because this was an unpaid project, it opened up the opportunity to use people who may not have necessarily had the experience on a game development team before this. Giving newcomers to the field a chance to cut their teeth and prove their worth is always a good goal with unpaid projects, and this was no exception.

So with that mindset present, the following people were recruited:

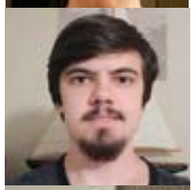


Kara Raya

A studio art major in studio art with a good deal of experience in modeling and 3D asset creation. While she hadn't actually worked on a team like this before, her portfolio of solo work was quite impressive.

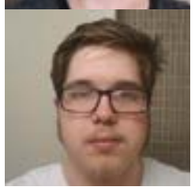
Jacob Wharton

He's the jack of all trades sort, who was very interested in getting involved in the game development world. He's had solo experience in coding, art, and design, so this would be his shot to really prove his stuff on a deployed project.



Logan Harold

He had hopped on and wanted to try his hand at programing. He was finishing up his CS degree and really wanted to throw himself at some game creation. He had a lot of experience coding, but none making games, so this was a good opportunity.



Taylor Lundy

Actually one of my students. One of the few who showed both interest in game design and actually came to my lectures on the subject consistently. He was down for anything we needed, and was willing to learn whatever, so that seemed promising.

There were actually a good deal more people involved with this at the beginning, or at least who wanted to be involved. They aren't listed here because at the end they fell victim to that inevitable process of discovering how hard game design can be and quickly realizing they were much better off playing games than making them. But that's just the nature of the field, especially for newcomers, so it's not unexpected.

With the 10 of us ready to start, we began to assign roles. The issue was that nobody wanted to be a lead programmer for this kind of deadline. It's to be expected, the plans at the time were to create a custom series of controllers and hardware that literally didn't exist before hand, then somehow get it working and deployed for the public in two weeks. Nobody that had worked on a game design team before wanted to do that. I attempted to convince Ethan Shelton of being the programmer for the project, but he –wisely- saw what hell would be awaiting him, especially with so many unexperienced people on the project, and insisted that he would be a backup programmer, because he also had real projects (read: paid projects) to work on rather than spend his time losing his mind on something like this. Again, I didn't blame him.

So we were set with a problem: anyone who knew what they were doing, and had enough experience to do it could see from a distance that this project was NOT something they wanted to take part in. So the answer was obvious: find someone who didn't have enough experience to realize what hell this was going to be, and give them the responsibility of main programmer. A bulletproof plan.

Enter **Sarah Hall.**

A CS student with an interest in game design, and happened to know Bryan Clark through a mutual friend. Bryan reached out and pitched her the idea, and she said she would be down for it, assuming that she wasn't the lead. The game was going to be programmed in Unity, which she had never worked with before and couldn't learn to the extent needed in two weeks.

....So Bryan brought her on and we made her the lead programmer.

Thusly with the team of 11 in place, we started off on figuring out 'what the hell should we make'.

The idea

The museum had an original idea of a soundboard like device that one of the curators had seen online as mentioned previously. It's a simple device with an ipad in the middle that showed footage from the actual mission. However given the sheer complexity of that board and our ever-closing time limit, we decided to pass on that. It also is to be noted that we are in fact, game developers, and so if possible we'd like to focus on making a game rather than a silly board with buttons in it. Or at least, we'd prefer to make a silly board with buttons in it that controls a game.

So after a few days and meetings we decided to create a sort of combination between the two. It would be a complex looking board reminiscent of the Apollo 11's main board, and it would control a custom game that we would create from scratch around our control panel. The ideas for the game itself were numerous and whittling it down was its own battle.

Fig B: Screenshot of the game "Affordable Space Adventures" for the WIIU

We looked at other games of a similar type, and decided to make something akin to the game 'affordable space adventures' where the controls and their complexity made up a majority of the gameplay. I showed the team a screen shot from the game (Fig B) as an example.

Split paths

After a good deal of discussion, it was decided that Keith and myself would create the actual physical board for the game, and the rest of the team would make the game part. Due to conflicts in the nature of prior experience in the game development parts, we ended up going down two roads.

Logan, Taylor, and Jacob all had experience in an engine called pygame, which is used for making 2D games in python. Ethan, Bryan, and Alex all had experience making games in unity. So we had a bit of an issue deciding which engine to work with for the project. I was confident that Keith and myself could make our board work with either of them, so we were fine with any choice.

After some deliberation, we decided to go for both. Logan, Taylor, and a few others who were interested in the project at the time would create a 2D game, while Sarah, Kara, and Alex would create a 3d version of the game in unity. Bryan would work on the actual gameplay of the game, while Jacob would figure out how to actually arrange everything on the screen in a ui that didn't want to make people vomit.

So we went to work.

Jacob and I talked at length about how to make something both historically accurate as well as entertaining/pleasant looking, while not making it too complicated for people who are just passing by.

We looked into some old interfaces at the time and found a few examples of what things looked like back then (Fig C). After a while we realized that we should go for a more fun, modern looking game that has reminiscent of an old technical interface rather than make it completely from the 80s.

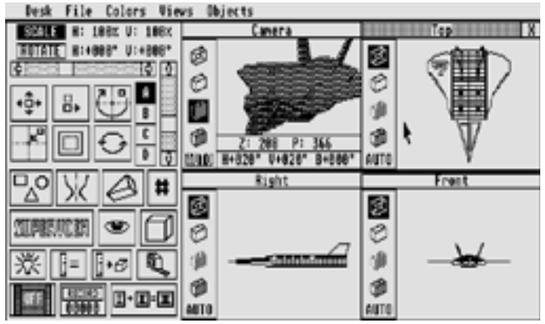


Fig C: An example of the kind of look we were aiming for with the retro/complicated look.

Bryan got to work making a game that would actually be both winnable and fun. Alex was waiting in the wings until we got some models for him to animate, and Kara got to work creating a 1:1 scale model of the Apollo lander itself.

I worked with the museum to compile a list of resources we could use as reference to our creation and sent them out to the team to use. From that point all of the individuals got into their own groups and hit the ground sprinting.

And they're off!

Fig D1: The podium after Keith had his way with it

I was able to convince the museum to graciously let us use one of their old pieces of hardware- and old wooden podium that had been hollowed out to allow for a screen- and get them to deliver it to Keith's workshop in town. As soon as they dropped it off Keith immediately went to work completely re-doing it and re-finishing it in a way that would suit our game's needs (Fig D1)

As expected of someone with his experience and expertise, he had the thing ready to go within the span of a few days, leaving the rest of us to catch up to his absurd pace.

I went online and found as many switches, knobs, sliders, and buttons as I could and ordered them with amazon prime, praying they would get here in time for us to put them together. I also ordered a bunch of wire and a few tools so we could make this device as fast as possible. Things were kind of expensive, but could have been worse. (Fig D2) shows the first shopping cart full of materials. This was before the actual electronics and circuit boards were ordered.

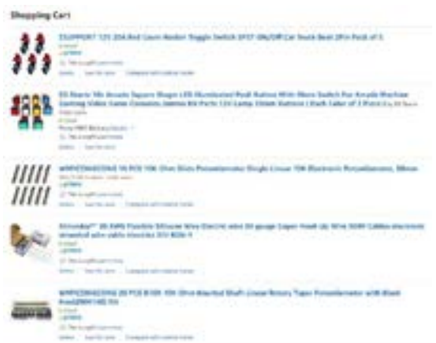


Fig D2: The cart filled with anything that would get here in two days and could be wired to something

At first the goal was to recreate to the best of our ability, the original Apollo control board (Fig D3). After the team got to look at that, we decided it would be best to maybe just go for something that looks vaguely complicated rather than attempt to recreate the literal thousands of buttons and switches present on the actual control board.

Once the parts were ordered, Keith's task was to create a layout that worked with what we had. 5 switches, 10 buttons of five colors, 10 sliders, 10 knobs, and a large on/off switch I ripped off of a mechanical device a few years back that looked cool.

There were a LOT of ideas thrown around at this point. So many in fact that we ended up filling several pages of concepts trying to narrow down what the hell we wanted the game to be and look like.

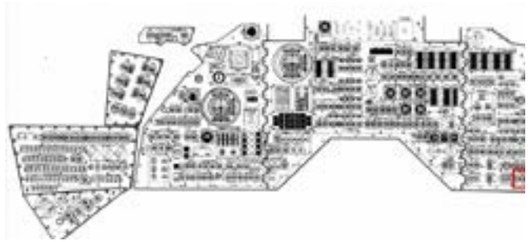


Fig D3: An image of the actual Apollo 11 control board

Meanwhile in the art department…

Fig E1: The initial reference image we used for the lander model itself

Fig E3: The first sketch

Kara and another one of the artists who started off with us were working on creating concept models of what they wanted to capture in the lander. There were a good deal of reference photos going around, and the slow realization that the lander itself was both massive and complicated.

The idea at this point was to still create two separate versions of the game, so the 2D team was leaning towards a more cartoonish version of the lander, as it was for kids. They found some reference art that captured the style they were aiming at in (Fig E1).

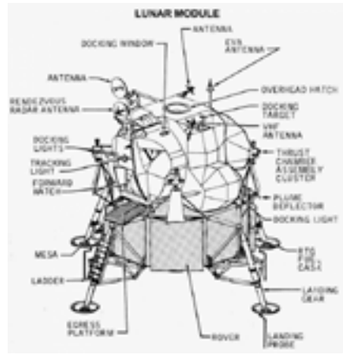


Fig E2: The schematic we used to base the art an animations off of

The art team realized at this point they would need a good deal more than just images of the lander to recreate it exactly as it was, so Kara had to delve into the Smithsonian site to find some actual schematics and technical documentation of the lunar module itself. At the end of it all, the reference image (Fig E2) ended up being the saving grace we needed to both get the art team on the same page as well as prep the animations and game design portions. Bryan and Alex got to work figuring out what specifically would be best to animate on the device, and how the game should progress as the player continued. Jacob took the schematic and started designing a layout that would reasonably be able to control it.

One of the original 2D artists, Keagan MacDonald, sketched out this variation of the lander to attempt and re-create in the 2D version of the game (Fig E3).

Back to the programming team(s)…

It was a bit chaotic during this stage for the programming team. With two separate versions of the project and over six programmers between them, people had to start stepping down and focusing on other aspects of the game. Jacob had originally been assisting in the development of the 2D game version, but there were too many cooks in the kitchen and he had to focus more on the user interface of the game. Ethan had also been attempting to assist with the programming, but with everyone else doing the same thing, he had to take a back seat and answer questions as they arose rather than lead a programming front.

It's important to note that at this point Sarah hadn't actually confirmed that she would be the lead

programmer, nor had she seen any of the project. So getting her informed and up to speed was a top priority, especially for the artists working on the 3D version of the game in unity.

Realistically the programming teams also didn't have too much to even work on at this stage, considering we still didn't know what the game would look like or do, and we didn't have even half the art assets we needed to yet.

And now we re-convene

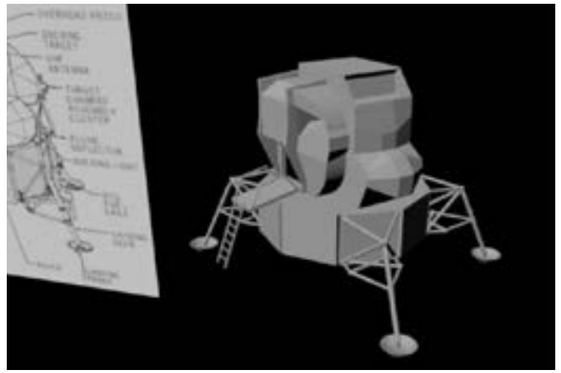


Fig F1: The first version of the 3D lander model

Getting the teams back together the next week was to share progress was an exercise in patience. With little programming done and the deadline coming up soon, we realized how badly we needed to focus on the design of the project. I wanted to take lead in that situation, but I knew I was too busy working with Keith and the actual hardware to step in when needed.

Both Kara and Jacob had made some excellent progress though from an artistic perspective. Kara had finished the initial shape of the lander model (Fig F1), and Jacob had plans for game interaction by then.

Once we got everyone together I ended up creating what would become the first rough draft of the game portion. (Fig J2) …and luckily one of the guest artists that had stopped by for that meeting was willing to translate my attempt at a layout into something a bit more readable (Fig J3)

The idea was that each module would be positioned around the view of the main lander, with some sort of instructions appearing below the actual module.

During this meeting we were able to get in touch with Sarah, who agreed to head the 3d programming side of things. It wasn't long until she had an actual demo of things changing colors according to button presses. (Vid 1)

After the meeting the two programming teams, led by Sarah and Skulling respectively, started discussing how they would make the games similar enough that each would work. The plan at this point was to pursue both games, and if one wasn't able to get done in time, we'd use the other one. And if both got done in time, then awesome we

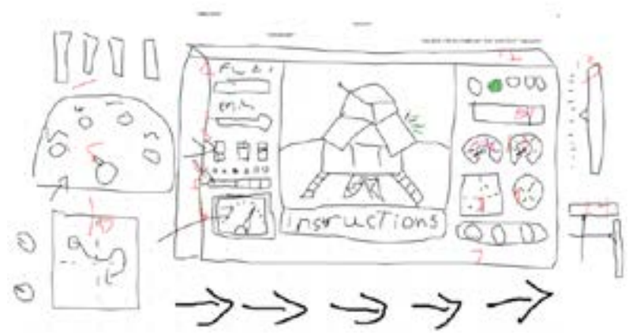


Fig F2: The initial game layout sketch

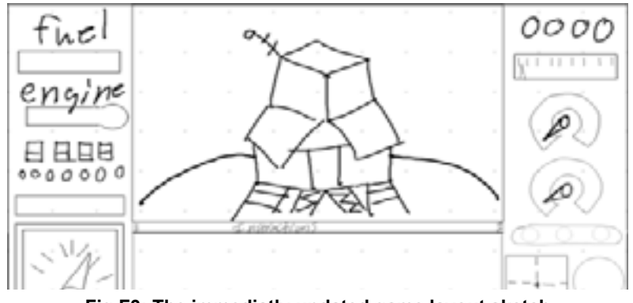


Fig F3: The immediatly updated game layout sketch

Vid 1: Sarah's initial button input code in action

have two games. Because the goal of this deployment was more for the experience than it was for the money, it was imperative to let as many people have a hand at making it as possible.

Designing a layout

Fig F1: The first actual sketch of the game screen

Sometime later Jacob released the first official sketch of our final layout. (Fig F1). It was made to be as historically accurate as we could get it while having it also be fun to play.

It was still unclear at this point if the two game versions would be sharing a layout, and which one would use which specific assets. It was temporarily agreed on to make the 3D version have a completely 3D layout and buttons, while keeping the 2D version exclusively, well, 2D.

Jacob had gone full ham with the designs at this point, and actually came up with the entire layout and game style out of nowhere. The entire concept was massive and spanned across several screens, along with a number of clever features that were sure to win us some brownie points with the museum staff as far as entertaining the guests was concerned.

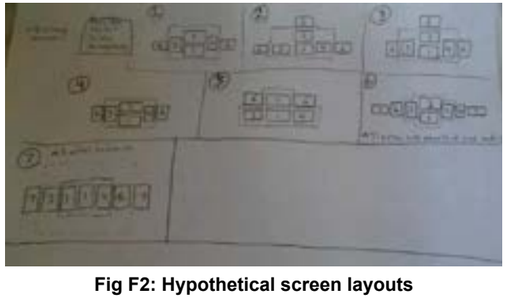


Fig F2: Hypothetical screen layouts

The idea was that each module was attached to a separate screen of control, with the main screen holding the content for the lander itself. This was of course, only if we had time.

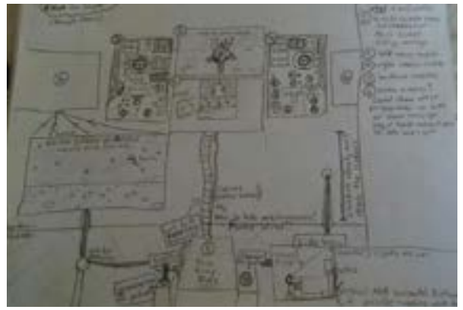


Fig F3: Layout idea for the screen contents

We knew the deadline was rapidly approaching, but so far this was the first solid grasp at what the final thing may look like that we had access to. So Jacob continued on and wrote up a bunch of notes for us to use and design the game part around. The notes applied to both if we were able to use multiple screens, or just using one.

Fig F4: Various screen augments and possiblitites

Progress is made

On the first of July, Keith released a mockup of the actual board. Which was fantastic, as the majority of us were stuck until we had an idea of what it would look like. Keith made an entire layout incorporating all 6 switches and 10 buttons for a whopping 16 digital inputs. He then put in 5 sliders and 12 knobs, for a horrific 17 analog inputs, all of which would (ideally) work at the same time. (Fig G)

This was the big starting point we needed to get everything started, both from art, design, and programming. Now we know what the board looked like, now we knew what we were making.

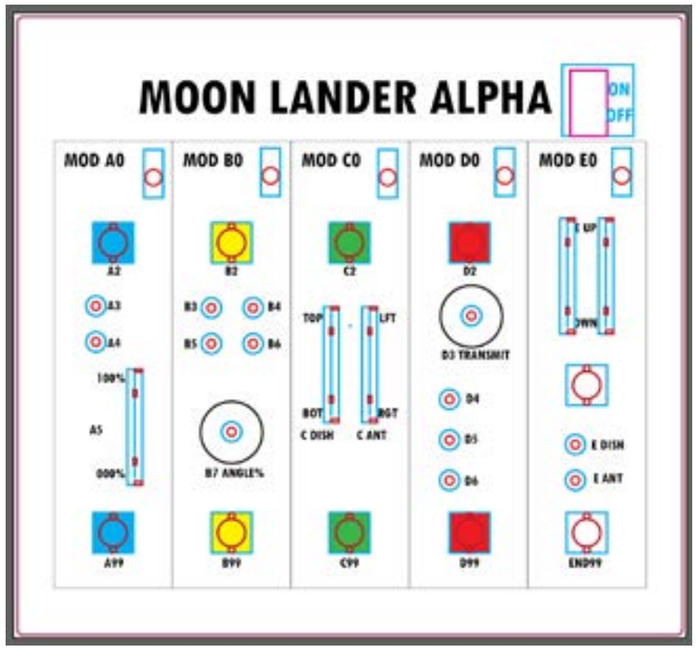


Fig G: The physical board layout to be sent to the lazer cutter

Of course with all major forms of progress there was a bit of confusion. Sarah was the first to realize that everyone was working on different ideas at this point, so we held a meeting to get everyone on the same page, which did wonders for our productivity. Not everyone was expecting the layout to look as it did, so a few programmers and artists had to pivot. The goal at this point was to get almost anything that works deployed, and use whatever art assets were made at the time to make it happen.

Another fun note was that Chris at this point mentioned that there was literally nothing he could do as far as sound design is concerned because we still hadn't settled on a game style just yet.

The physical gets real

Now that the actual layout had been decided on, Keith and I had to redouble our efforts. Keith secured a laser cuter and materials to print it with, and got to work sanding and creating a version of the board to fit the podium.

I had to start working on making the actual buttons themselves. The goal was to make some clever looking buttons, knobs, and sliders to be used on the actual control board. But to do that I needed to 3D print a lot of material.

Fig H2: The podium itself, with the cover and buttons added

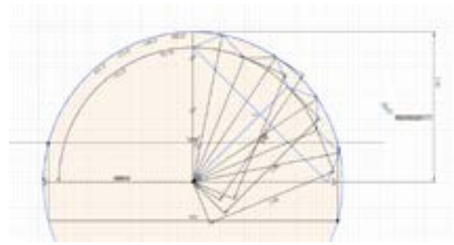


Fig H1: An autocad screenshot of the internal knob mock-ups

I started with a few sketches and after getting as close to the measurements as I could of the switch headers, I got to work actually modeling what the prints would be. It was a huge issue because the degree of accuracy required for the knobs we were using was around 0.01 of a millimeter. There was an unholy amount of math and lost sleep trying to figure out the precise angles that would need to be printed to fit perfectly on all 12 of the knobs, much less be so sung that kids wouldn't wander off with the caps. (fig H1)

Luckily I was able to stop by the engineering department at FSU and get some assistance in getting the exact measurements of the tiny knob angles with their specialty tools. Kieth recommended using glue to secure the caps, but I was against it as we may have needed to swap them out over time depending on what the design for the game became.

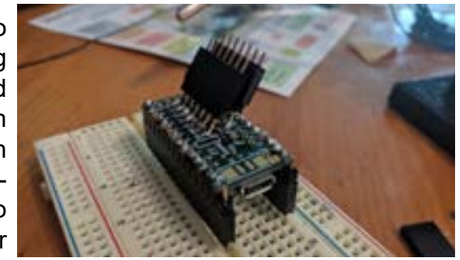


Fig H3: The miserable Teensy board with all of the extra joints attached, placed on a breadboard in Keith's workshop

It wasn't long until Keith had the entire board printed, mounted, and all of the controls secured to it. It was a sleek and streamlined thing, and the controls were as smooth as butter. (Fig H2)

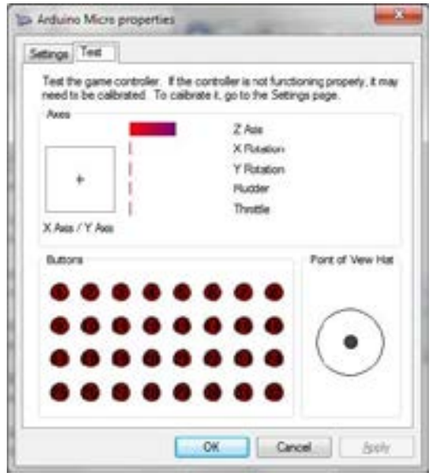


Fig H4: The initial game sketch

Now we started to run into the issue of how on earth to get this many modules to fit on a single circuit board, much less into windows to recognize. I bought a Teensy and gave it to Keith along with a schematic of how they could theoretically be wired, and Keith quickly discovered that it would mean adding custom pin outs to the board. He went through like 5 soldering iron heads dealing with that nightmare. But the final product had enough inputs to where it would work, even if it was held together with hot glue. (Fig H3)

While Keith took care of the physical part, I started working on a set of drivers to work with the teensy that would be able to actually read in that many button inputs. It was an abomination, but I was able to get the teensy to not only register as a circuit board, but also a usb game controller with an absurd amount of button inputs. (Fig H4)

Design time

After seeing what the actual board looked like, Bryan delivered his own rendition of a control module for the screen. It was... a little complex (Fig G1). But the important part was that he had noted which games do what, and what controller would affect which device.

While he was doing that, Jacob had been doing research into the actual control module of the real lander, and had some excellent reference material for me to use as I attempted to create the physical knobs for it. (Fig G2)

Fig G2: The origional control board

Fig G3: The specific knobs in question

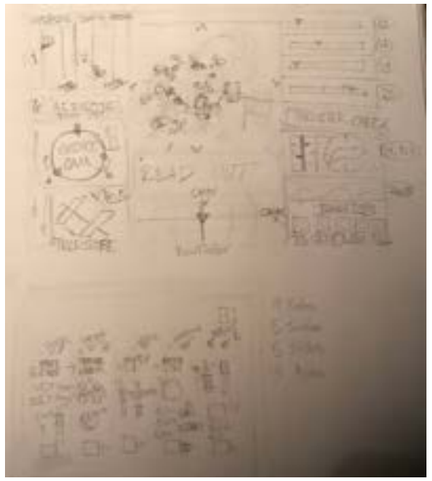


Fig G1: The sketch relating the games to the physical board

The original idea was to make as many knobs look like the actual module knobs as possible (Fig G3), which were standard beak-knobs with a notch in the top. Getting the angles right would be imperative and the rounded-yet-angular shape of the knobs themselves are what really gives them that old-school feeling.

Let's see how 2D is doing...

While we had been working on all of that, Logan and Taylor had been hard at work creating the two dimensional version of the game. We got our first look at the design attempt there. (Fig H1). They were going with Jacob's idea of having multiple monitors control various aspects of the game. I had a stockpile of monitors lying around and enough hardware to hook them up to almost anything, so I was down with that plan if we had time. As always, the project deadline was an ever-present Sword of Damaclese hanging above our heads as we continued attempting to get everything done in time.

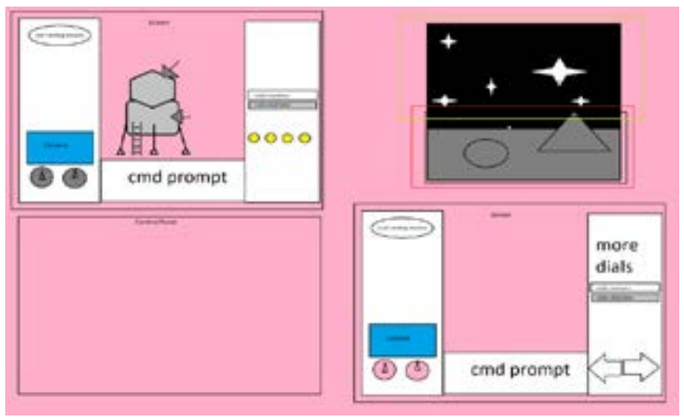


Fig H1: Various layouts for the 2D version of the game, including a view of the actual lander and the space around it

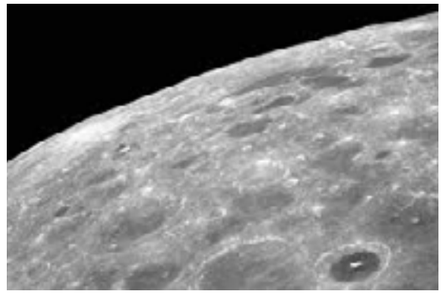


Fig H2: The backdrop we decided to use for the game

The 2D team had also been working on getting a skybox that would fit with the rest of the game, and found a few choice stock images to us to use. Free use, of course. (Fig H2)

During all this Kara was able to deliver the completed Lunar model. All that was left is the textures. (Fig H3)

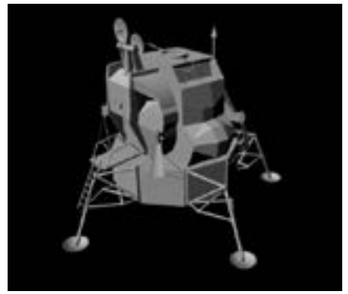


Fig H3: The finished moon lander model with details

Pain and suffering

So it turns out that the default windows controller emulator can't actually display the amount of inputs I was trying to mash into it, and while it had no issue with the digital inputs (it could handle over 125, which is amazing) the analog inputs were proving to be incredibly challenging. Each of the 18 analog inputs had values from -32,000 to 32,000, meaning 64,000 possible values for every single knob. That's 1,088,000 possible values that needed to pump trough a single USB. USB of course, being a four pin connection where two are ground and power, leaving me a whopping two data inputs to feed those values through. The 16 digitals were easy in comparison, each could be either pressed or unpressed, which means they could be one of two values. So 32 possible values to feed there. Not hard at all, consdiering windows could go up to around 128 if pushed.

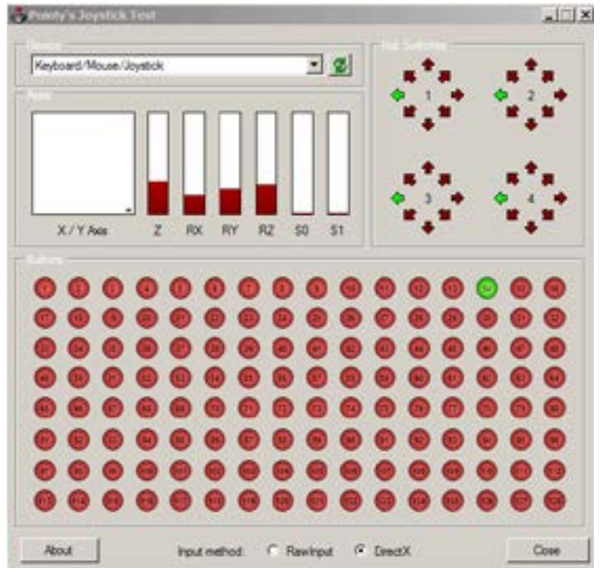


Fig I: The game controller appearing in "Pointy's Joystick Test" program.

I ended up having to use a program online called "Pointy's Joystick Test", which was the only utility that actually displayed more than the standard 6 axis windows was comfortable with. Thanks to that and some code from a crazy guy who had made his own version of "how many axis can I put on a single circuit board before my computer has a stroke", I was able to get the basics of the board working with the HATs.

It was about this time I discovered that without any actual input on the board it doesn't register a proper ground and just kind of has a seizure when there isn't anything soldered into the slot. That made testing super fun. (Fig I)

But after several hours' worth of suffering and maligned drivers, I was able to get a functional test working on the circuit board where windows registered it as a game controller and spit it's inputs back out to the screen. It was a marvelous day of celebration. (Vid 2)

Vid 2: The first time the inputs actually worked on the computer. At least, mostly worked.

Hardware fun

Keith went through and wired the entire board, which was a task I can only image involved copious amounts of some sort of painkiller given what it took to wire all of them together. All of the buttons and switches had at minimum 3 wires from each, most with far more than that once the ground, positive, and data connections from each were added. Not only that, but there was another massive circuit going between the main ones that held 12v so the buttons could light up when plugged in. (Fig L1)

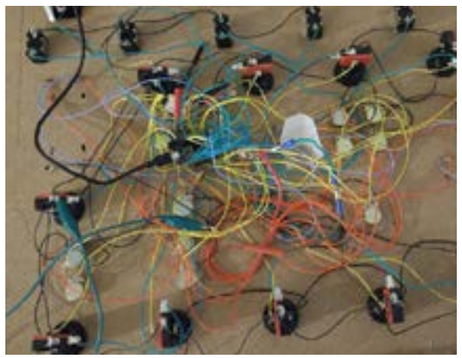


Fig L1: The rat king of wires that Keith was able to create to actually connect it all together

He dropped it off at my workshop and I got to work actually wiring it into the teensy itself and the computer. There were a few mistakes here and there, but after testing each button and input individually I was able to re-wire the few missing cables and get them all working on my Unix box. I'm amazed he was able to keep all of the wires separated and connected through all of that, I'm still convinced some kind of dark ritual must have been involved.

Fig L2: My additions to the already suffering Teensy board

There were a few major issues with the way we had to put the board together though, one of them being that the sliders we were using weren't meant to be used in a board this thick, so Keith had to come up with some really creative ways to get them to stick. There was a lot of hot glue.

Another issue occurred when I realized that the teensy itself couldn't supply enough power for all of those, so I had to wire in an extra power circuit with its own ground. (Fig L2) that sucker was by far the most solder-covered, hot-glue-stained, miserable circuit board I've ever worked with. After me and Keith had done our part to that thing, it was screaming for death. But it worked, which is all we could ask for.

The deadline approaches

Now that we were 10 days out of when we needed to have the game finished, everyone was in full sprint mode. Keith went up to Montana to see the Bozeman computer museum to get some notes on how their displays function and what ideas we could use from that, and was incredibly helpful. (Fig J1)

Fig J1: Keith being the saint he is up in Montana

Bryan took Jacob's designs and turned them into something more reasonable given the fast-approaching deadline (figure Z) It was incredibly complex, but given that we had a controller with 18 axis on it, that was to be expected. He came up with ideas for each of the five games, as we had decided to break the game down into five modules and code them according to the module's color.

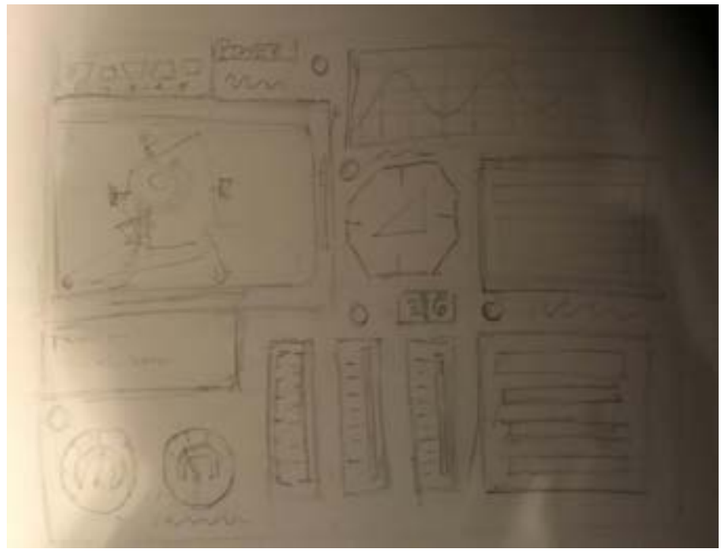


Fig J2: The new layout for the game screen

Jacob started focusing on creating as realistic buttons and switches as he possibly could with our current design, and ended up taking a bunch of frames from the actual lander module itself. (Fig J2-6)

It quickly became obvious that we weren't going to be able to make every single module reproduced exactly, but we still needed enough to make the player think that there was more going on than there actually was on screen.

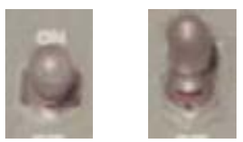


Fig J5: Round switches

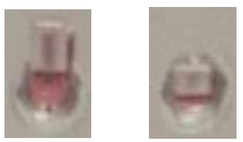


Fig J6: Flat switches

Fig J4: The various displays and sliders from the control board

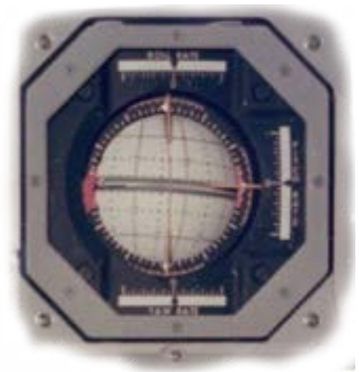


Fig J3: Axis control module from the actual landing console

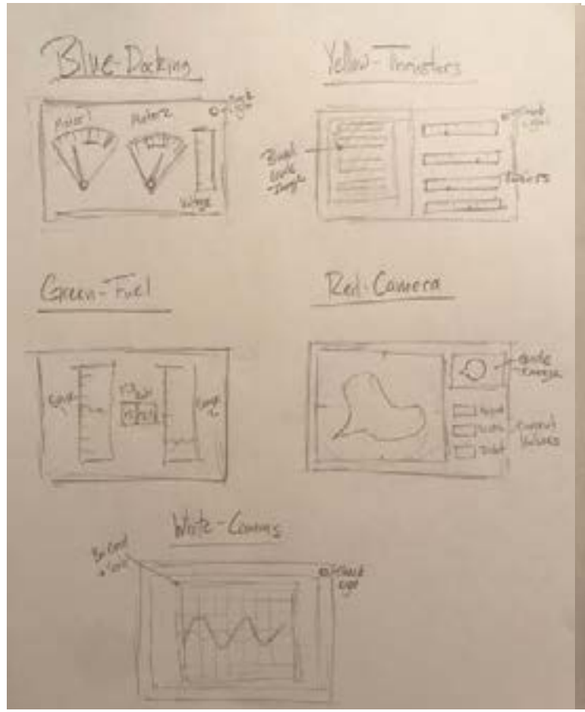


Fig K1: The new sketch for how each game would be played, in relation with it's position on the physical control board

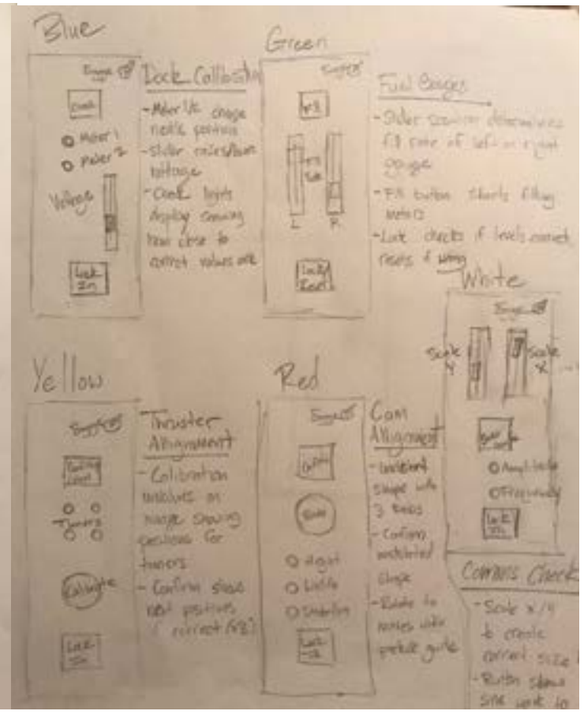


Fig K2: A cheat sheet describing which button and knob relates to which function in the game

The actual games

There were…. a lot of issues leading up to the actual game completion at this stage. We were 7 days from deployment, and Bryan had just finished the actual design of the games. (Fig K1) It obviously took him a while as he couldn't really get started on making the games themselves until we knew what the actual board looked like, and that was a process in and of itself.

Luckily he was able to get the games designed and laid out in time for us to start mapping it to the actual board. He also created a cheat-sheet of what each slider and button would do in-game (Fig K2), which is what we desperately needed to get started programming in earnest.

We had a meeting with everyone and went over what all of the games did, and it became apparent we were going to be REALLY short on time if we were going to make this happen. We still hadn't decided on an actual mockup of what the games would look like on screen now that we knew what the games would do.

It was officially time for the big guns to come out.

The big guns

Six days before deployment, Sarah reached out to Ethan requesting assistance. She had been essentially not sleeping trying to get all of this done, and was making absurdly amazing progress considering she had literally just learned unity two weeks ago, had never made a game in her life, and had no access to the actual board we were using. But her inexperience was catching up with her and there was only so much she could do without causing physical harm to herself given our resources.

Ethan and Sarah started meeting up to divide what code they had and try to split it in a way that would actually lead the game to being competed in time. This caused more issues though, as Sarah had somehow done all of this on her own with no prior experience up until this point. Because of that, she didn't have time to implement any form of programming standards, and hadn't worked on a team of programmers like this before either way. So her code was… messy.

Really messy.

Up until this point that hadn't mattered a bit considering she had somehow done the impossible and gotten this far without any outside help. But now that Ethan had to take a look at it, it was causing problems. Ethan has made games before, and worked on large teams. He was used to following coding standards and keeping things clean and in line. For him to jump in at the eleventh hour and try to start salvaging this was a bit of a herculean task.

It lead to both Ethan and Sarah re-writing each other's code multiple times. Because of the chaotic deadline, we didn't have time for code review and so things were everywhere. It's funny, because little changed on the outside of the game, but the code itself was rarely the same codebase for more than 12 hours at a time. During the day Sarah would do what she could, and at night Ethan would come on and do his part.

It was…. Rough going. But in the end it was enough that the deadline looked like it would actually be met, at least from the programmer perspective.

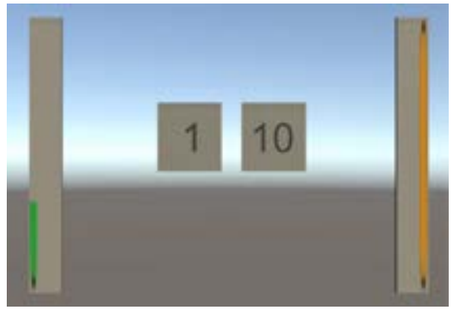


Fig M1: The game representation of the two-part fuel system

Style and art

With the deadline on the horizon, we all began the age old practice of sprinting till our fingers bled. Kara came up with the first on-screen image for use in the game, the part of a fuel gauge. The Apollo 11 used a two part non-combustable fuel system and we had been trying to replicate that visually. (Fig M1)

The idea at this point was to use the artwork being created for both of the games, 2D and 3D. This was the first time we'd acknowledged that we wouldn't be able to make the buttons on the 3D version in 3D.

Kara was working on those art assets due to the fact that she couldn't actually texture the lander until we got a solid grasp on which of the parts would be animated or moving. Jacob was able to get some public domain images that reflected the actual stuff on the lander, and then used Photoshop to separate the layers. The idea was to create a photorealistic version of the lander that would actually reflect the real control module. (Fig M2)

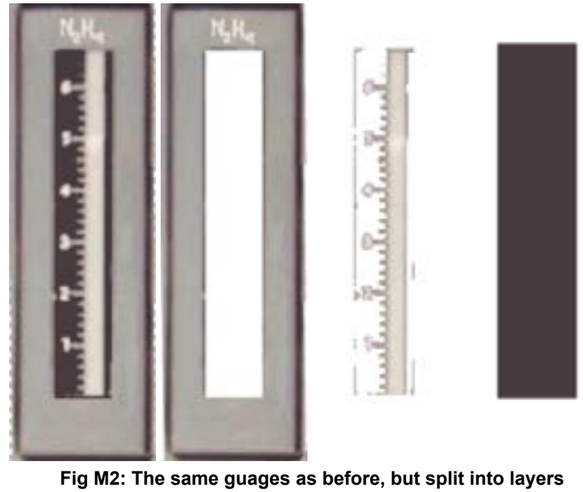


Fig M2: The same guages as before, but split into layers

There was a good deal of debate back and forth on which style we wanted to use, pixel art vs photorealism, and it was hard to decide without seeing the actual layout of the buttons on screen first. Jacob was in the camp of pho-

torealism, as the product was for a museum. Kara was trying for pixel art, as it made it seem more like ‘game’ game.

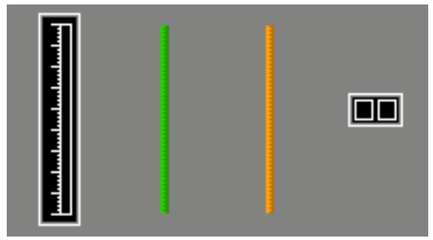


Fig M3: The green game sprite sheet

After a lengthy meeting discussing the merits of each style, they decided on going with pixel art as it would be easier to create given the timeline. After that the first sprite sheet was born, this one for the green game (Fig M3)

Fig M4: The screw

Jacob got to work on creating the actual aesthetic and other background elements of the game, trying to keep it both pixel/cartoony without getting too caught up in realism. Soon he made the first screw, which ended up being a basis for the rest of the game’s art. (Fig M4)

The screen layout- or lacktherof

Making a layout for the actual screen proved to be a bit of a bear. While we all knew what the game would sort of look and play like, it was up in the air as to how we would actually lay it out specifically. This was double in part due to the fact that with no budget, there was no screen for us to use. The people at the museum said we could use one of their IPads, but I vetoed that on the grounds that it would be too small. Instead I asserted that I would find a screen for us to use, one that was of a more suitable size.

I then completely forgot about that promise until launch.

Jacob had a tough time making the layout due to the fact that there wasn’t actually a set resolution, so I just threw out 1920x1080, as that is the standard of HD monitors. As long as we made the layout scalable, there wouldn’t be much issue. Even if we weren’t using vector art, doing anything with pixels would look more or less the same if blown up or shrunk down.

It was rough assuming any non-wide-screen layout. Jacob posted the following words of wisdom with the first screen breakdown; “Let’s assume we only have 1024x768. Then we can still make large modules like 384x256, 256x256, 128x256, 192x256, etc. And 64x64 would be more for a smaller piece of a module, and honestly the gauges could be as tall as 192px in some cases... this is if we had 7 modules, but we are only doing 5 or so, so we could even realistically do larger.”

The trick was to make the screens appear as busy as possible, like an airplane cockpit. We didn’t want people to sit down and think it was even remotely easy to pilot the lunar lander.

All the progress

With an idea of a layout, and more importantly an idea of what each game would be, Kara went to work creating sprite sheets for the rest of the games. The blue game was next (Fig N)

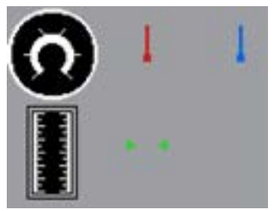
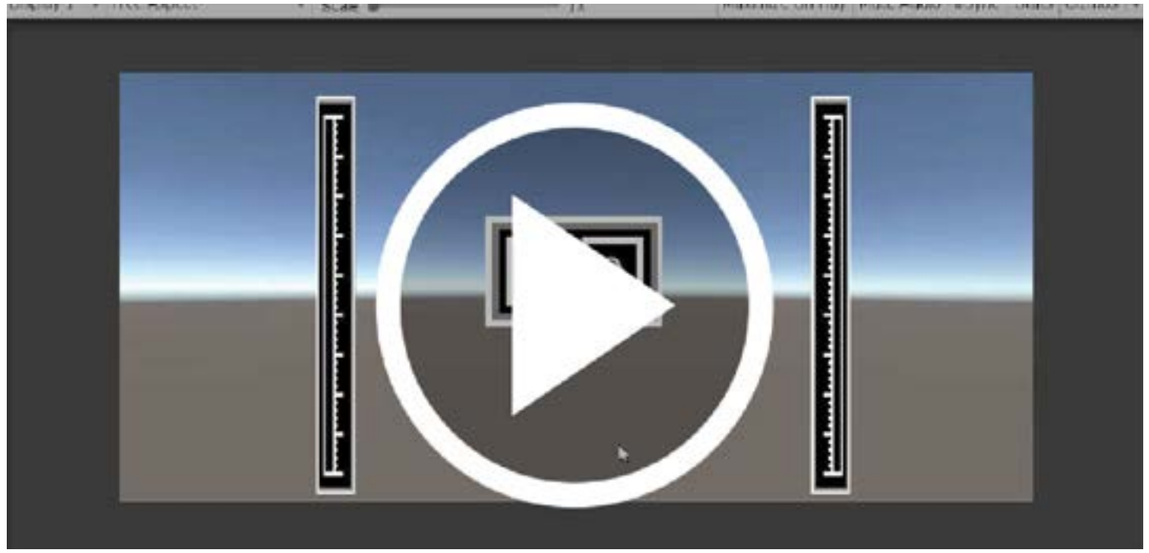


Fig N1: The blue sprite sheet

The yellow game was after that, which was a series of images played after one another to hide the actual content of the game until the user used the large dial to find them.

Sarah was able to create a working variation of the green game at this point, and after she got the game itself working, was able to add in the first of the sprite sheets from before to see what they looked like when the game was played by a user. (Vid 4)



Vid 3: The green game with the sprites added

Throughout the entire process Bryan was working with Sarah and making sure the game itself was both fun and winnable. They ended up deciding that a ‘goal area‘ would be ideal for the green game rather than the flat numbers. Ethan came through with a few variations of indicators, because there were a lot of ways to layout the ‘goal’ of the sliders game, especially considering the numbers changed to rapidly.

The red game was up for debate next, as it had a few issues with its concept. The original idea was to distort a shape and flip it around to make it work like the shadow of the shape in the background. Each of the three knobs would control one of the axis, and the large knob would rotate. The problem is that was WAY more challenging to program and we only had a week left.

Logan had actually created his own variation of the red game for the 2d version, as the 3d plan we had wouldn’t work in a 2d environment. He created a variation that essentially stretched and skewed a 2D image, using the dials. The issue with that is there was no efficient way to make use of all four red dials in a single 2D image.

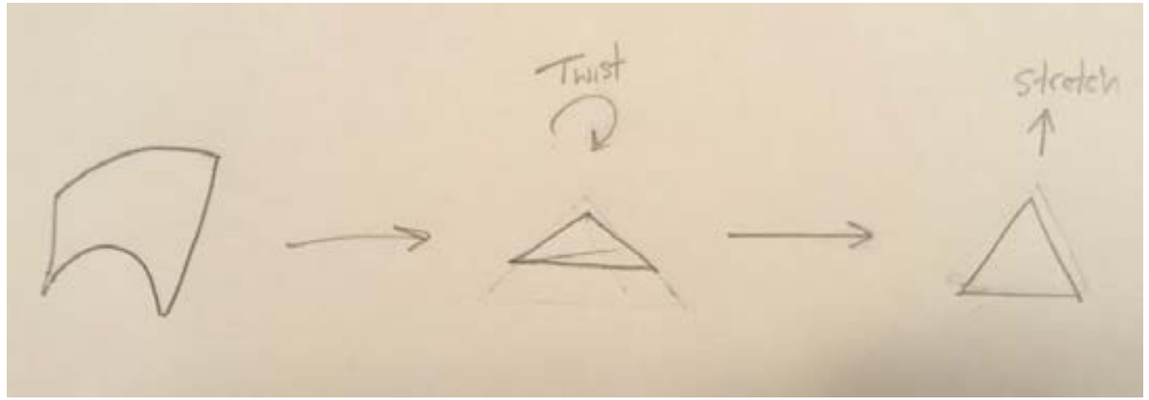


Fig N2: The original plan for the image to be distorted in the red game

Bryan had a plan to (Fig N) do the similar style of distorting the triangle, but it would be 2D. The problem again, was that there was no way to use all of the dials with a 2D shape. There were a lot of attempts to give each of the dials their own axis for which to distort the shape, and we briefly planned for the unity 3d game to have a different style of game than the pygame version, but we had to acknowledge there just wasn't time enough for that.

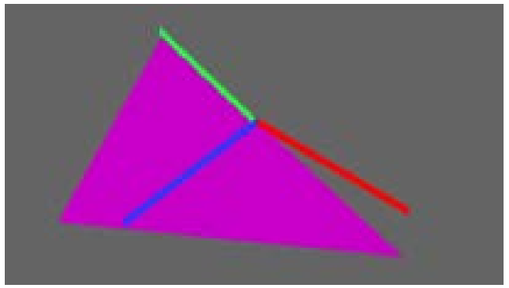


Fig N3: The 2D version of the red game, where each of the three lines is controled by a seperate dial

In the end Ethan created a game where you are presented with a triangle of a fixed shape, and each of the knobs controls one of the three lines, with the large knob controlling the origin point. It wasn't what we had in mind, but it worked out really well with early testing. (Fig N3)

This was enough for Logan to recreate the game in it's entirely in pygame. It was the first of the color games that had been created in pygame before unity, which really put the teams on par.

Sliders and knobs

While everyone was working with this, I took a break from attempting to get the circuit boards working due to immense frustration, and focused on getting the buttons printed. Turns out 3d printing those suckers was going to be a bit of a bear, the likes of which were summed up in my messages on July 8th at 3am:

"Just printed the first attempt at buttons.

Total disaster.

BUT I learned what not to do."

Fig O1: The blue slider model

A few days later on the 9th I was able to actually print working ones, which meant the next step was deciding on a design to go with. The blue slider was by itself, so it needed something larger to take up the extra space. The most obvious solution to this was to just make it a sort of arrow, so it could take up both it's own groove and the spot next to it. (Fig O1)

Fig O2: A rounded slider, made to look sleek

Fig O3: A fatter slider, easier to grip onto

The other two slider sets on green and white were both up in the air, and we couldn't figure out a good design for them. I attempted printing a few variations of slider, seen in (Fig O2) and (Fig O3). In the end I was running out of time, and after a few variations of each, realized that the more cylindrical slider just plain felt better, so I cut the side off of that and printed it for each side (Fig O4).

Over the course of the next few days I was able to design and start printing and fitting the remaining knobs. I came up with four separate versions of the knobs that we could use, and asked the team which one we wanted.

Fig O3: A fatter slider, easier to grip onto

The first knob was based off of a standard soundboard dial (Fig O5). The second was an attempt to re-create what it looked like the original lander had, at least from the blurry images we were able to find (Fig O6). The third was a way to give it a sort of grip on the side to make it easier to turn (Fig O7). The last one was a bit of a bother to create, as the inner stock had to be large enough to keep the metal dial in place, while not encouraging kids to just pull on it (Fig O8).

Fig O5: First dial

Fig O6: Second dial

Fig O7: Third dial

Fig O8: Fourth dial

The team's responses were all over the place, and nobody agreed on which to use. So I ended up using all of them, as only four of the games had knobs. So each game would have a separate dial shape and layout.

After all of that I was able to print the final knob (Fig O9), which was to create a scale representation of the knob used on the ship (Fig O10). It took a while, but I was able to get the measurements almost as exact as we could hope for off of the images we were working with.

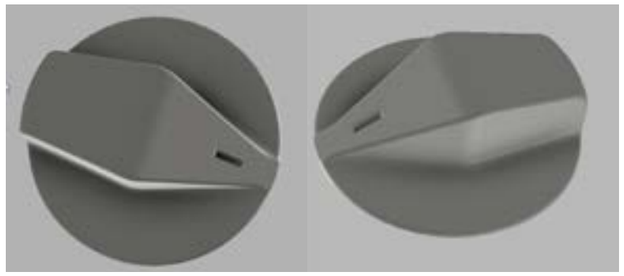


Fig O9: The dial model

Fig O10: The dial it was based off

The yellow game

The original idea behind the yellow game as Logan came up with, was to have a blurred image that slowly becomes clearer as you slide the dial around. The issue is that blurring the actual image took up way too much processing power, and became a bit of an issue on the 2D version. Jacob suggested a solution to this by making a series of obstructions to cover the content, and only having certain parts be visible as the dial rotates.

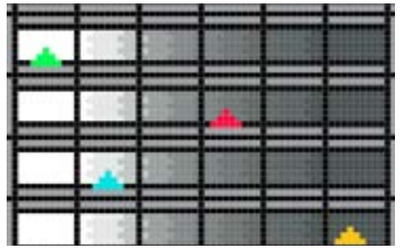


Fig P1: The initial draft of the yellow game

That ended up being what both games did as soon as they got the concept down. The goal was for both of the games to have a 'correct' layout, where the game would register sucsess as soon as all of the arrows matched each other. Kara got to work making a series of sprites to match the concept. (Fig P1) was the first iteration, then moved onto (Fig P2) to make it a bit more clean looking.

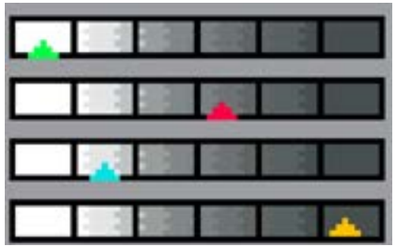


Fig P4: The updated version of the yellow game

She came up with the idea to hide each of the arrows with black bars, and made a series of sprite sheets that kept all but a few hidden depending on how the dial was moved around.

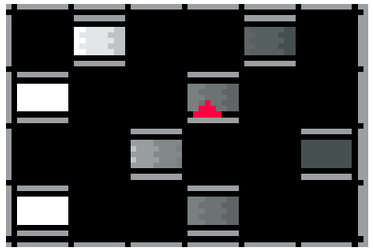


Fig P3: The game in action

At this point Taylor got hit by a car (not a joke) and had to step out of development for a while. So the 2d side became completely housed on Logan's shoulders.

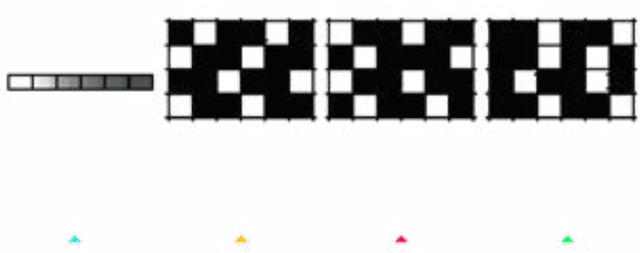


Fig P4: The yellow sprite sheet

This was a bit problematic, as at this point skulking still didn't quite understand the concept of the yellow game. It wasn't until the unity side created a working demo that the idea got across to skulking that there were in fact, two separate panels.

Eventually a solid concept was drafted and Kara created the final sprite sheet for the yellow game (Fig P4)

4 days left

We then had a whopping 4 days to actually make the game work. I still hadn't gotten all the knobs printed correctly –designed yes, but printed no- the 2d game only had 2/5 games working, the 3d game had 3/5 and no winnable state… or working state. The games hadn't even been combined onto a single project yet. Birds had been making the simpler games on one build, and Ethan had been creating the more complicated ones on another. We had no sound effects, no idea of what the sound effects would look like, no textured art, no animations, and were overall in over our heads.

At this stage we had to cut losses. We had a lengthy meeting with the 2D team to gauge their progress. It turns out that the original artist who was going to make the lander model…didn't. And with Taylor having been hit by a car, skull king was on his own. It wouldn't have been so bad but he also hadn't been able to attend several of our previous meetings on the games, and wasn't even sure what half of the games looked like.

So after a long talk and much deliberation, we decided to scrap the 2D game. There were only two people left on the actual 2d project at this point, so skulking hopped on sound and Taylor got to work on button art. It was a shame, as Taylor had finished the green game and Logan had the red game working, but there just wasn't enough time.

Sound

Logan immediately realized that we had literally no idea what sounds would be in the game, as Chris had kept telling us, and that just now donned on the rest of the team how bad that was. I made a sound design chart template and sent it over to skulking, who then went through and detailed every sound he could come up with and where it would be used. He stayed up all night and most of the next day working on it (Fig Q), but he was able to finish it enough that Chris could come in and start making the actual effects.

Chris, true to his nature, was prepared for this level of crunch time as he had worked on games with

Fig Q: The massive sound effects list and instructions

myself in the past. He buckled down and spent the next three days doing nothing but foley work for the project. He had to create some weird sound effects and ended up getting a travel mic and driving all over town with it smacking random things with other metal objects. He even made a contact at the junk yard who helped him make the engine sounds with of some old cars.

For the fuel sound effect, he literally recorded himself gassing up his car. He looked like a crazy person there with the gas tank and a windscreen, but the sound came out great.

It was a bit of an interesting goal, considering that some of the buttons clicked on their own, while several of them were completely silent. Matching each of those to one another was its own task. Our goal had been for every button to make a separate sound when pressed, but the programmers didn't end up having time to implement that. Somehow thanks to Chris' diligence, we ended up with more sound effects than we even used in the initial game.

The layout?

With 2.5 days before deployment, we still didn't have a layout. Jacob had a few sketches but nothing digital yet, and Bryan had been focused on the actual gameplay. Taylor was going to help with the layout art, but because he was still recovering after the car incident, Jacob had it all on his own plate.This was a huge blow for the programming team, as without a layout they had no idea how to combine the games, or in some cases even how the games were supposed to work.

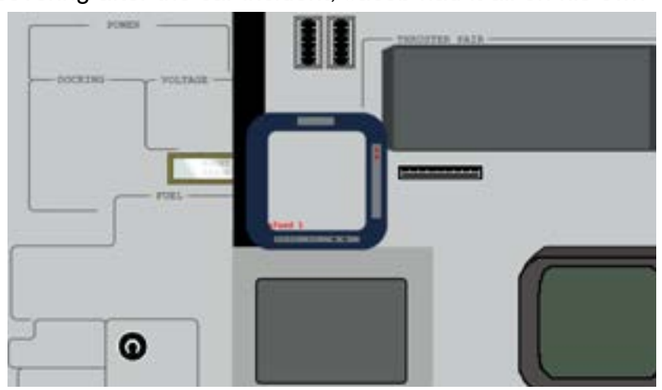


Fig R1: The first digitial version of the game screen layout

At the night of the second to last day, Jacob gave us a rough layout to start basing the game itself off of. (Fig R1) It was rough, but it was enough that the programmers could actually get started making a game out of it.

The next day, the day before launch, Jacob gave us version two of the layout

(Fig R2). It was much more complete and looked to be exactly what we needed to get the game running. It was missing some of the more complicated aspects of it, but it was pretty good for the first draft of what would go into the game.

Because the layout actually had spots for each of the five minigames, Ethan was able to take his designs and immediatly put them into a usable form in the program that he had been putting the current minigames into.

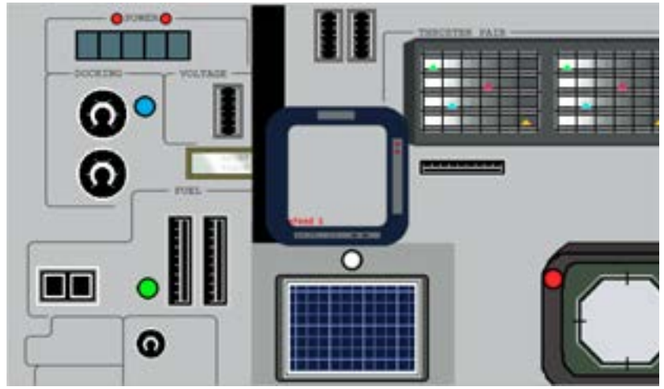


Fig R2: The updated game screen, ready for use

The crisis

We were unmistakably behind, but everyone was pretty sure they could finish their parts by the time it came to deploy. During this time I had been working constantly trying to get the joysticks to function in a way that made sense.

I had them all working on Unix, but for some reason any time I attempted to load it into windows, it could only recognize 6 of the axis. All of the buttons registered fine, but the axis were the pain point. I had hoped it was a unity issue, but after Sarah spend some time messing with it she determined it was most likely something in windows.

I had her try to get the game to work on a Unix environment in the meantime as a fall back, as I knew I could get the axis functioning if I had access to bash. Sadly that wasn't able to happen as she had never installed any kind of Unix on an ASUS motherboard before and didn't have the distrobutions on hand to do so. I set up a private server for her to download them from, but even after that she couldn't get it to load on her computer. She tried getting any version of linux to run, even mint, but was unable to get Unix working on any computer she had. Without a way for her to test the game on unix, she had to compile it and send it to me to try and run on one of my machines.

That was when we discovered that unity REALLY doesn't like unix environments, especially not without some serious leg work.

The cavalry

So seeing no other way out, the cavalry was called in. I reached out to Ethan, Techno, Wired, and every other programmer I had worked with that might know something about this. We had a lengthy meeting in which we went through every conceivable option trying to figure out how to get windows to run a joystick with more than 6 axis.

Turns out it wasn't actually my drivers, it's actually built into windows like that. Windows literally cannot take input from more than 6 axis on a single usb. That pretty much shot the project where it stood. Without a way to get the main control panel working, there was little we could do to make the actual game work.

It was a sad, sad day/announcement, but we didn't have a choice. We had planned for all sorts of failures, but not this. If art hadn't come through, we had backups in mind. If programming wasn't able to deliver, I had a contact that would be able to finish it. If we didn't have the sounds, I had a hotfix and library we could use. Even if the controller board didn't work, we had a backup in mind. There was no plan for a complete inability for windows to recognize the joysticks. Even if I had programmed

it in differently, it still comes down to 18 axis on a single usb, which was a hard stopping point for not the code, but the operating system itself.

That's not to say we didn't give it our 110%. Missing the deadline was the biggest no-no possible at this point, so I came up with some wacky solutions to the issue in my attempts to solve it last mintue.

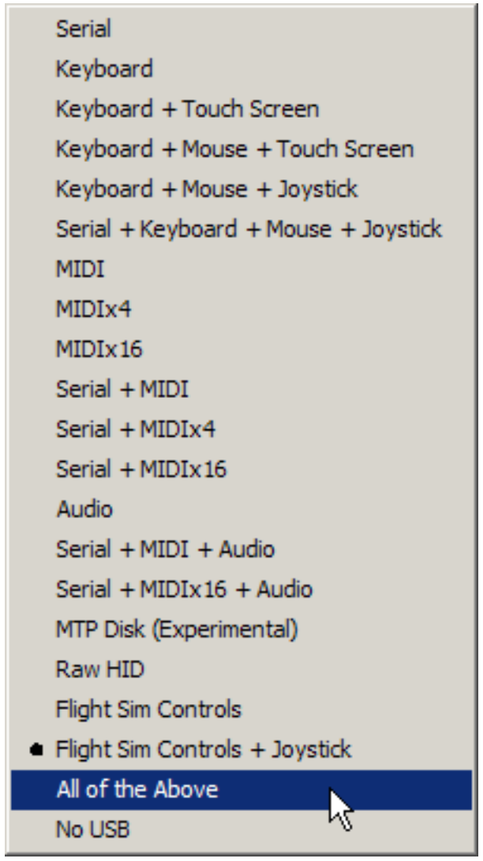


Fig S1: Top ten photos taken seconds before disaster

Wacky solutions

Fig S2: The new device as windows saw it

Drivers are a complicated thing, as I discovered. But due to their complexity, they also have a surprising amount of customization to them. It's very possible to build a driver to be almost anything, and have windows register it as all sorts of crazy stuff.

Upon looking through the options for the default framework libraries that the teensy would use, I found that not only were they completly editable C libraries, but also have a large number of templates to work from that are built into windows.

While you can use it for the common things like MIDI, Keyboards, Joysticks, Mice, etc, there exists a suspicuos option labeled "All of the above". (Fig S1) This option, when selected, from what I was able to gather, attempts to cram every single driver combination into a single USB device. I have no earthly idea what possible use this could have, but considering we were already struggling trying to get the acis to fit onto a single USB, it stood to reason that if we somehow put *more* drivers into the board, it may work better.

This was hilariously wrong, of course, but not without some incredibly entertaining results before hand. I labeled the device as "All The Things", which windows had a stoke attempting to register. It first attempted to label it as a keyboard, (Fig S2) before loosing it's mind and repeating the driver instillation on a loop for a few rounds before giving up and considering it 'installed' (Fig S3).

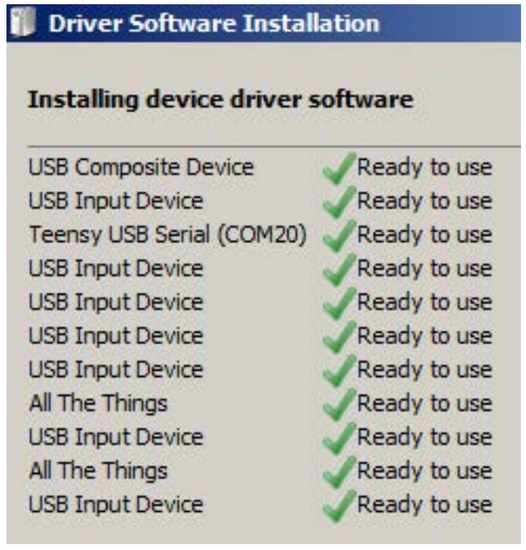


Fig S3: Windows trying it's best

Attemtping it with the joystick test software from before was it's own oddity, as windows couldn't figure out what on earth the device was attempting to register as, so it just gave this mystery USB a single button (less than a standard mouse), a single x/y axis (no joystick uses this), and no hat functions (meaning it had no

Fig S4: The program attempting to parse this mess

buttons somehow, even though it was being registered as a keyboard). (Fig S4)

To add insult to injury, I then went back into the C libraries it was using for drivers and attempted to change a few things while forgetting to remove the device first. This corrupted the connection in it's entirety, resulting in a driver title that had several chinese and unicode characters in it (Fig S5). I had to completly wipe the Teensy to get it back to normal, and I still have no idea why someone would ever use the "All of the above" template.

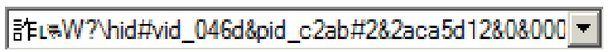


Fig S5: The updated game screen, ready for use

Swing and a miss

In the end though, because we didn't have the knowledge/time/drive/skill to start messing with a kernel, we had to call it off.

This was the announcement sent to the team that night: "Regrettably we were unable to find a solution in time for tomorrow's deployment date. The project, sadly at this point, has failed. I'm going to call the representative at the museum and ask what alternatives there can be."

Sending that email was disapointing to say the least. But it was the Sunday before our deployment, and we had to face the music and accept that despite best efforts, we aren't capable of getting around a flat out hardware limitation. I sent the email and made the announcement to the team to about as much excitement as was expected. I immediately got calls from a few team members that hadn't been at the meetings asking how on earth this wasn't possible, and that there was surly some way to make it happen. I had to explain to them that regrettably this was one of the few things we hadn't counted on, and was very much beyond our control. I immediacy ordered another bunch of teensys on amazon, because there was still a plan B. Or I guess something like plan G in this case, because we had a LOT of possible solutions that didn't work prior.

This delay wasn't unexpected on the museum's end, apparently. Turns out they were aware that somehow getting a piece like this done in the span of two weeks is more than a little ambitious.

Fast turnaround

After a little back and forth with the museum, and a whole lot of apologies on my end, they agreed to let us miss the opening, but still deploy. So we set the deadline for one additional week, raised our standards, and started sprinting. It worked out well in this way anyway, as Ethan was going to be completely unavailable following this week, so we didn't have much choice but to deploy anyway. We couldn't really stretch the deadline anymore without a complete programing team.

After the initial disappointment, the majority of the team was actually pretty relieved we had the extra time to deploy. Chris was able to finalize his sound effects and finish off some of the more niche aspects of the ambiance, Jacob was able to put some additional touches on his layout, Bryan was able to get with Kara to start working on an additional layout now that there was more time, and

Sarah and Ethan could work on actually combining the games. It turns out that at this point, all of the five minigames were in their own separate files, and nothing had been combined. Which meant that winning the game, any of the power settings, and all of the combination layout hadn't actually be even implemented yet, which was terrifying to find out, but just made me all the more glad we had another week.

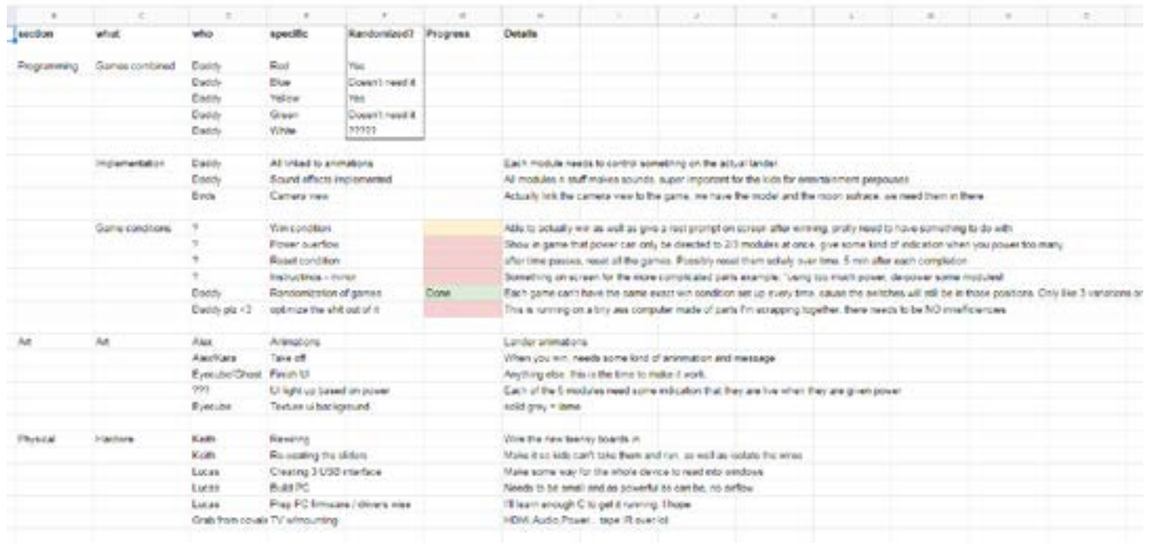


Fig T: A screenshot of all of the features we planned to add in our last desperate week of development

The task list for the final week was…. Also very ambitious. (Fig T) Considering we did now have an additional 7 days (which was essentially 30% more time in the grand project scheme), we wanted to add a few meaningful but easier features to implement to make up for it. It also gave us a chance to address some previous shortcomings that we kind of botched in our haste to deploy in time. There were…. A lot of issues we didn't realize.

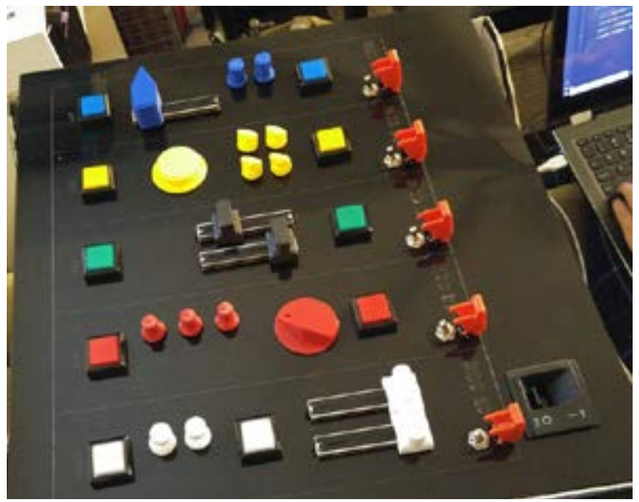


Fig U1: The board in all it's glory. Additionally, Sarah working on programming it off screen on the right.

Physical failures

So the actual board itself, while looking nice (Fig U1), had some more serious flaws surrounding the wiring, the podium, and the dials themselves.

The red and yellow dials I had 3D printed came out without their bottom layers actually printed, which made them appear hollow (Fig U2) The print settings for this project were all over the place, and I had just been planning on putting the hollow dials in temporarily then swapping them out all sneaky-like after I printed better ones. But with additional time I could actually print good ones to start with.

I ended up just re-slicing them from the ground up in hopes that it made them print the remaining layers, which thankfully it did. I also had to adjust a good deal of the slicing settings, and a brim wouldn't work for the base of the knobs because there was no way to sand it enough to get the actual metal knobs to fit inside the caps otherwise. (Fig U3)

Fig U2: The bottom of the Red knob that didn't print correctly

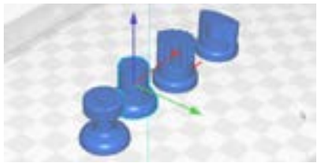


Fig U3: The new knobs in the slicing tool

It also finally occurred to the team that the entire PC would be running in a closed-off sealed wooden box for 10 hours a day for 6 months. Whoops.

Keith was the first to both realize and address this issue, and he was able to secure a light up PC fan to actually put inside the box facing outward towards a grate, so it could get some minor air circulation in there. (Fig U4). After some initial testing we realized it effectively cut the operating temperature of the pc in half, which was exactly what we needed. There was no budget for this project, so it came down to "What does Lucas have in his horde that can fit in this wooden box". Technically I could have ordered parts and made a nice computer specifically for this purpose, but I was already in the hole several hundred up until this point, and missing the deadline meant I sure as hell wasn't going to ask for any funding. Luckily my horde had a good deal of broken PCs, and I got started trying to put one together that would be the least dysfunctional when deployed for a long period of time. (Fig U5)

Fig U4: Keith's awesome fan solution, that terminates to a standard motherboard fan power jack

The more major issue however, was the damned sliders. Turns out the sliders I ordered weren't actually deep enough to get through both the podium of the box in addition to Keith's amazing acrylic. Keith had come up with some creative solutions, but we quickly realized those were going to have to be augmented if we wanted this to last long term. The wires for the sliders, up until now, had been shoved through tiny little holes and hot glued into place. That was the only thing keeping the sliders on. Three wires (hot, ground, data) and a dollop of hot glue. Needless to say it caused some serious issues long-term, especially with kids messing with it.

Fig U5: The pile of PCs to choose from

The most obvious of problems was the fact that one of the sliders not only didn't work, but was also very hot to the touch. After some basic sleuthing it turned out the issue was that the cheap fraying wires I bought for the project off amazon were actually connecting the hot to the entire metal slider casing. So that was both crazy dangerous and dysfunctional. After talking with Keith for a bit, he had the idea to instead of having the sliders go directly into the acrylic, possibly creating a raised platform for the sliders to rest inside of that housed the actual wiring and electronics. This was pretty brilliant, and it worked out well. Keith had some similar solutions in the past, and was able to cut out some phenomenal styled acrylic specifically for that purpose. Once the teensys got back, I gave him back the board and he got to work on rewiring them.

The plan to get around windows

The remaining issue of course, was that no matter what we did, windows could still only handle 6 axis per usb, regardless of how well programed the teensy boards were. We discussed several possible solutions to this problem, including things like going to unix, constantly swapping inputs to trick windows, and even having a game mechanic where only certain sliders were active at once.

In the end I decided the best bet was to just do what windows wanted and only have 6 axis on a usb. This of course didn't work with our design, which had 18 axis, but luckily didn't matter. I was just going to make one usb into three.

So the new plan was to program three separate teensys to different usb slots in the computer, all registering as joysticks (players 1 – 3), and have each of them control 6 axis and a section of the buttons. That way we could still use all of our knobs and sliders, and windows didn't have to deal with more than 6 per usb.

The rewiring for that was pretty atrocious, and I'm incredibly thankful Keith was able to make that happen. The programming for that was somehow worse. I had assumed it would just be like making several controllers, but the way we had the board set up meant that there were a different number of buttons for each circuit, and they all had their own ground and hot, which couldn't be run across the separate circuits. Not only that, but the analog ports available were different for each teensy, as we had already wired the first one and the second had most of its ports filled with buttons. That meant each individual teensy had to have its own custom driver to get it to function.

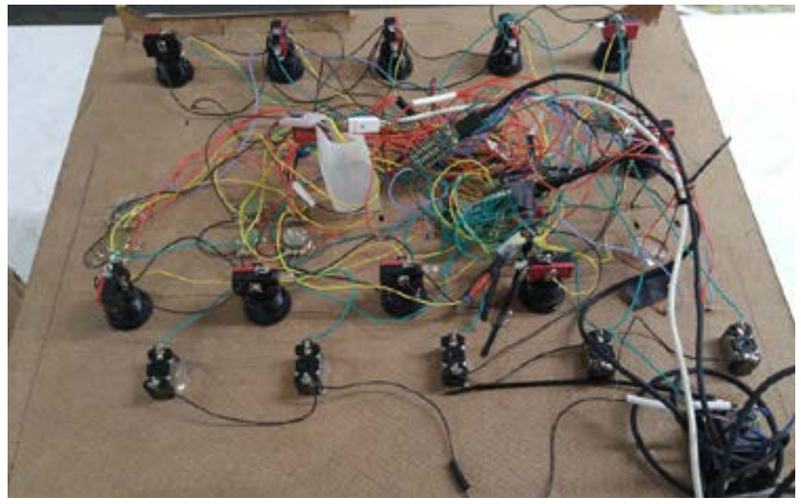


Fig V: The new, somehow even more complicated, wiring

To make matters even more complicated, unity was being very picky (in part due to me establishing the drivers on linux and expecting them to run on windows, oops) about which controller buttons it would give to each actual teensy. So teensy 1 had to have its axis 1-6, while 2 had 7-12, and 3 had 13-18, even though the axis were wired to different ports on each one. So the computer needed to recognize them all as different controllers initially, then once it got past windows crap it would then synch them up and start treating them as a single controller.

I could go into detail on the mess it ended up being, only compiled by the fact that it didn't occur to me to actually color code the teensys (which all looked the freaking same) until much later anyway. But suffice to say it was a miserable and bloody battle (Fig V). The point is, this had a much higher chance of working than the random stuff we had been attempting before, so we were going with it regardless of how horrific it may look. God help us should we ever need to add anything else into this miserable board.

Sound stuff

Chris, true to his nature as a foley artist, made a lot of sound effect variations. It was hard to get the idea across that there weren't really 'contact' sounds happening in the game, as much as just buttons that played sounds that would make people think of the actual lander. Obviously in space there wasn't any sound to be heard, but we still needed to make the users feel like they were making progress.

He ended up going through each and every one of Logan's notes and making several variations of each sound effect, to the point where we had to create a separate FTP server just to store all of them. Between the signal generation and effects processing alone the sound files were massive.

He also had the idea of including actual recordings from the Apollo mission itself into the game, which turned out incredibly well, and were set to play as each module was finished. He sent this message to the team one night: "It occurred to me that this is a public job, and that all of Apollo 11's mission transmissions were recorded, and made public in the 70's. Why guess what is sounds like, when I can just use the real thing!"

More programming

The initial steps in the new deadline were to assign everyone tasks so nobody stepped on each other's feet, as well as get everyone up to speed.

Sarah was about to be waist deep in finals week, and had already been falling behind on classwork due to teaching herself to code and somehow creating all this in the span of two weeks, so we agreed she would probably be best to wean off of it for now and let Ethan take the lead, as he's used to the miserable sprints that occur during these projects. Luckily they both agreed and we got started.

Ethan's first goal was to complete all of the remaining games, then re-do them in a similar fashion so they would work with each other, then combine them into a single game. After that he'd need to line up all the animations and put in the sound effects, which it turns out still hadn't been added to the game.

Ethan would also create a way to randomize each game so that when the player won, they weren't in a default win state again when a new person game up. That's the issue when dealing with physical knobs and switches. Once you move them, they stay that way in the real world. Unlike a controller where the joystick snaps back to neutral positions, the switches, sliders, and knobs stay once moved. So that means that each of the five games would need to have a series of different solutions that cycle randomly so the player doesn't autocomplete their game when they start.

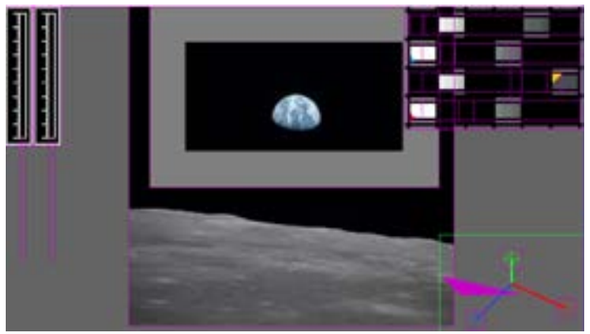


Fig W: The final stage the 2D game was in before we scrapped it and moved to focus on the 3D version.

Sarah hopped on the task of getting the camera angles to work, which was had its own set of challenges. The problem was with the initial design of the white game, Bryan wasn't able to come up with a use for the two sliders. The solution Logan and Jacob came up with was to have multiple camera angles of the lander and moon, and have those be controlled by the sliders. This worked out well because the sliders on the white game are the only module that has content above the final button location.

The issue is we hadn't discussed this whatsoever up until now, so Logan had to spend some time getting Sarah up to speed on how those would work. Never mind the fact that it also had to be translated from the original variation that was made for the 2D version Logan was working on originally.

It's funny looking back, because that was the first time we got to see an updated version of the 2D game since it had been scrapped. It turns out they were farther along than the unity version at the time, but we were so disorganized we hadn't realized it. Logan already had the yellow and red games working, and Taylor had finished green. (Fig W) It actually worked out pretty well with the concept of multiple camera views, and they had a few working mock ups of it. Ideally the sliders would just slide and pan between multiple shots of the moon that theoretically could have come from the lander…. assuming they had nicer cameras back then.

Github freaks the hell out

So as we were all sprinting to try and get this done, there were a number of weird and unexpected minor issue occurring. Github for one lost its mind and due to one of us trying to fix a few issues on the cloud, we somehow maxed our bandwidth and had a hard lock put on our account.

This would have been disastrous normally, but it turns out github had been having issues like this for a while now, and Ethan had already started swapping over to the unity collaborate project, which ended up saving us. Though it did mean that Chris couldn't put sounds in on his own, he would need one of the programmers to do it.

Another problem was that Ethan had been just stealthily working I the back end up until now, and now that he had to create all of the games and re-code what Sarah had done, he realized he had no idea how several of the games were supposed to function. So there was a lot of back and forth as I attempted to explain the games until I also realized I had no idea how they functioned. Up until this point I had pretty much left the game design to Jacob and Bryan so I could focus on the hardware and control board. There were more than a few instances in chat when Ethan and I both came to the conclusion that we were making the wrong game.

Design coming together

The extra time had given Jacob a chance to create some fantastic art pieces to bring the UI together. That, mixed with Ethan putting the games in, made a huge difference. For the first time in the entire process, it actually looked like a game. Like, with all of the modules in and whatnot, it looked real. And it looked *good*. (Fig X)

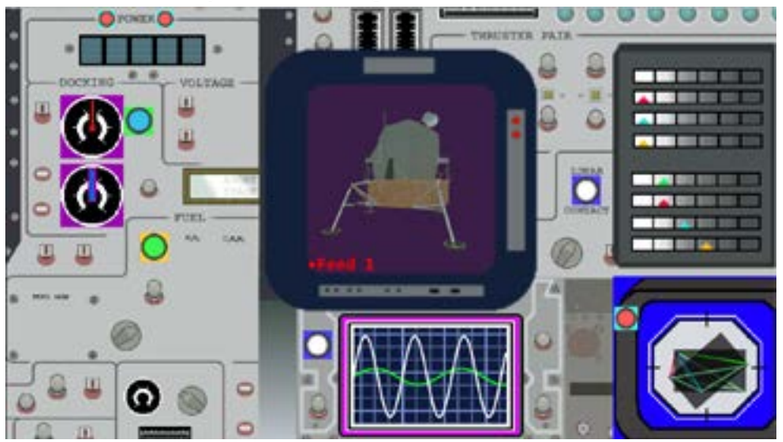


Fig X: The first working variation of the game we got to see

Alex had been adding the animations to the lander throughout all of this, and had done a pretty good job considering his distinct lack of instructions. The idea was that each of the five modules controlled something visible on the actual lander itself. Antenna, thrusters, ladder, etc. We didn't really care what was what, as long as things moved around for the kids playing the game. Just something that moves.

He had been working on those and was just now finishing up, which was perfectly in time for Ethan to start implementing the animations. Kara also had to come in and reiterate that the lander had exactly *one* good angle to look at, and anything else would shed light on its ugly polygonal face.

Because of that, Alex had to take a few liberties on where the animations worked and some of the positioning of the actual ship parts. For example, technically the satellite was on the other side, but it wouldn't have been visible from the current position, so he had to get creative and pretend it was actually facing us.

MORE KNOBS

Keith sent me some measurements for the new sliders, to which I was a bit perplexed at first, as I already had the slider knobs created. It wasn't until he dropped off the actual board that I saw what he had figured out.

So it turns out that when Keith created the new mounts for the sliders, he went all the way and made sure that no matter what a child did, they weren't dismounting them. He created a multi-layer custom acrylic case for each slider pair that had its own mount and glue attachment, then a slot for the knob to be printed into surrounded by felt to no debris could jam up the sliders. It was *fantastic.* (Fig Y)

Fig Y: The phenomnial looking board Keith created, with the new fixes for the slider issues from before

The issue of course is that with only three days left before we deployed, I now had to make all new mounts for the sliders. Which is a trial and error process by nature. So I took to the calipers and just started printing as much as I could. The idea was that I had a lot of filament, but not a lot of time. So I'd rather get 5 prints attempting variations of the same thing, in hopes that at least one would fit. Then I could head to the museum and swap out the bad one after we deployed.

This strategy worked, the only hiccup was the mass amount of sanding needed for the sliders to slide across the felt without getting stuck. I grossly underestimated how long it would take to get those smoother. A mistake I will never make again.

The plan comes together

After that it was off to the lab! The goal now was to check all of the wiring, and make sure the custom drivers were actually working for the damned board. The way the board was laid out, there was no rhyme or reason to which axis or button goes to which teensy, and no time to re-wire them to attempt and fix it.

Keith had done a brilliant job of making a temporary podium for testing purposes out of an old cardboard box from a grill, and I took all of that to the lab and set up shop for the day printing and wiring as fast as I could. It was the first time I got to see the button lights working that Keith had made, and it was really something special.

Fig Z1: Working on the controller board in the lab, rewiring with a volt meter and hitting the knobs with a mallet.

There were a few issues with the wiring that I had to address, as some of the grounds were too close to the hots, and if you jimmied the switches on top too much the inputs would occasionally corrupt, but it was enough to get a baseline down. (Fig Z1)

Sarah came in after her classes now that finals were ending, and was able to debug her code through the board, and make sure the switches and buttons actually related to something on the screen.

I had to use a mallet a few times to get the buttons on, but at the end of the day from a hardware perspective, everything was working aside from two switches, 3 knobs, and one of the sliders. Considering how massive and complicated that board was, especially now that it had 3 separate circuit hubs instead of one, I was willing to call that a victory.

Sarah was able to get them all responding one way or another in unity with the combined version of the game, in a way where none of the buttons or sliders actually conflicted with each other, which was a herculean task by itself. Because Ethan had combined them and was remote, he wasn't able to program in the actual hardware for it. He only had a keyboard, so there were no analog options in the game yet. Sarah had to sit through and convert each and every one of them from Ethan's keyboard testing controls to the actual board controls. It was … rough.

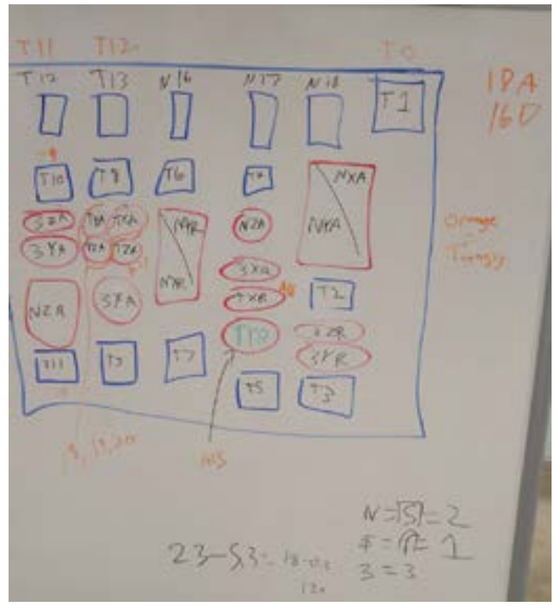


Fig Z2: The borderline cryptic schematic we had to make to keep track of the connections

In the end, the only way we were able to actually re-do the controls in a way that worked was by physically writing out which of the anolog and digital pins went to each of the teensy boards, and then drawing the physical connections on a whiteboard. (Fig Z2) We started to identify the teensys based on how mulched their individual usb cords were. N was new, T was twist tied together, and 3 was the third one. Not the best naming convention, but we were pretty tired at this point.

The final road block

The original plan had been to deploy on that Sunday. What I didn't expect was for the amount of programming needed to be done once I got the board working. Ethan was still making the games, but without a way to actually test them on the board, it was completely up to Sarah. She took the board home from the lab in its mostly-working state to start re-creating the games in a way that synched with the actual control module.

I realized very quickly that no matter how good Sarah was at this, she had literally just learned unity a few weeks ago, and the custom drivers I made for the board were less than user-friendly. While I feel like Ethan probably could have figured it out as he's seen my miserable chicken scratch on projects like this before, he was gone for the week as he had said. That means the programming team was just Sarah and caffeine. The solution to this issue was two part.

Part one was getting more time from the museum. This was an issue because we were already behind schedule by a week, and there was no way they'd give us another week. So I did what every great director does when pressed with a situation like that. I made something up.

I sent an email saying that the game was done, and it just needed a TV and we'd be ready. I said this knowing that the museum didn't have a TV, as they had planned on using an ipad to display the game, which obviously wouldn't work with our new design. So I sent the email planning to stall as long as possible to give the programming team as much time as possible.

The second part of the solution was getting Sarah some help. The majority of programmers I knew either hadn't been involved in the project up until now (and would take far too long to get them up to speed on this complicated mess of a game) or were unavailable as it was literally the last week of classes. So I called in a backup I hadn't used in a long time, in the form of someone who was already intimately familiar with the game. Our animator, Alex.

The thing about animation is that it's not a single-source skillset. The art is important, but so is the actual implementation, which requires programming. A good animator has both a solid grasp of the artistic side of things in modeling and textures, but also has a handle on the implementation and programming side of the finished product. This made Alex the perfect candidate to ask (beg) to hop aboard at this final week sprint and assist us.

Alex is actually quite good at programming, and because he's worked in game design before, is used to the stress of a final sprint. He was reluctant, as he finds his home in animation much rather than any kind of programming, but he understood the necessity of getting this project done and out the door as soon as possible.

Art upgrades

While all of that chaos was going on with the hardware and programming side of things, the art team used this time to pump out as much content as they could. Kara had finished the texturing of the lander at this point, and put the new prettier version into the game. (Fig AA1)

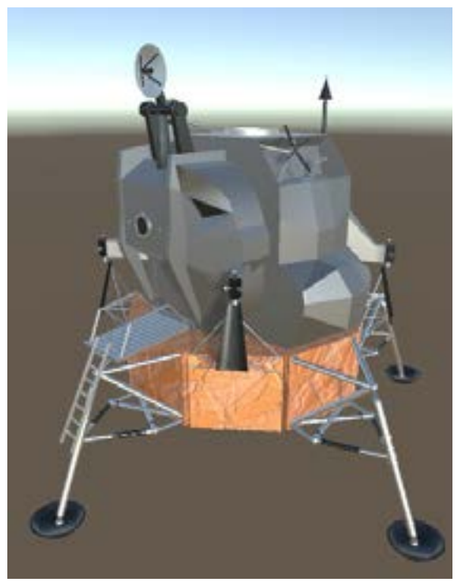


Fig AA1: The final model of the lander, textured and complete.

We also got a bit of a surprise with the layout at this stage. It turns out that because we had the extra time, Bryan had worked with Kara in brainstorming and was able to create a new layout for the entire game that matched more with the pixel-y style the games had become. This was a bit out of left field, but after looking at what they had come up with, it was undoubtably more in-line with the current game's theme. (Fig AA2).

The only immediate hiccup with that one was that I was still incredibily for the idea of having the actual lander screen be as large as possible. On Jacob's original design, I had made it a bit of a joke asking him to make the lander's screen larger. Kara's very first draft reflected the fact that she knew I would inevibly ask the same thing, and included a little note letting me know just that. The version was pretty slick for a first draft, and it also left room for all of the buttons and knobs on the actual console.

Fig AA2: The new, more cartoonish layout for the game, along with a note letting me know I wouldn't be able to convince them to make the lander screen larger

Once the games were actually in the layout, Ethan was able to create a sprite sheet (even though he was out for the week) with all of the different assets so unity only had to load in one large image instead of several small ones, which saved our loading times. (Fig AA3) We hadn't been noticing any specific slow-downs up untill this point, but considering we were about to attempt and run it on hardware that was made with a budget of "look what I found in the trash", we were happy to have all the optimization we could get.

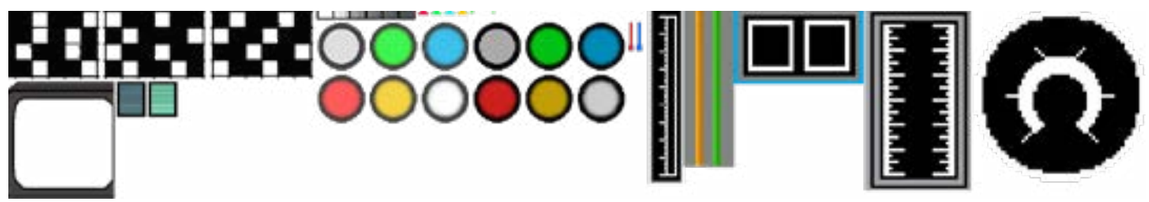


Fig AA3: The final spritesheet that made it into the game

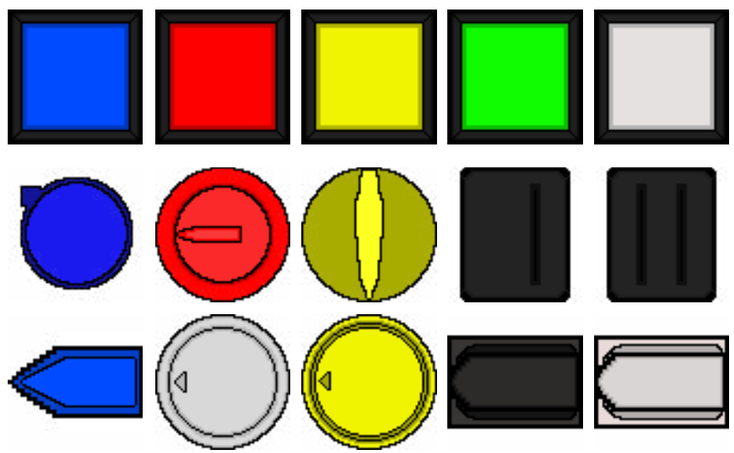


Fig AA4: The control panel sprites that were overlayed on the screen

Taylor continued working on providing the pixel assets for the knobs on screen. (Fig AA4) The idea was that for every button or knob on the control panel in the real world, there would be a representation of it on the screen that mimiced it's actions. He was able to create each not based off of any actual art, but of the images of the finished buttons and knobs themselves, so they ended up being extreamly accurate. This includes the fact that I ran out of green 3D printer fillament and had to print the green sliders in black.

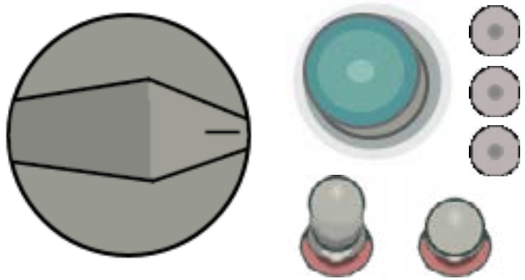


Fig AA5: Jacob created more cartoon versions of the knob, lights, buttons, and rivets present on the actual lander for the new layout we were using

Jacob was also able to finish up the minor details like the switches and rivets in the borders of the game screen itself. (Fig AA5)

Now that the layouts and the specific details were finished enough that we could see what they would look like in game, Kara was able to go back and add in the finishing touches to the final layout. (Fig AA6) This last pass through included things like light-up boarders for each of the games that would signal if it's been completed, as well as implementing all of the additional sprites Jacob and Taylor had created.

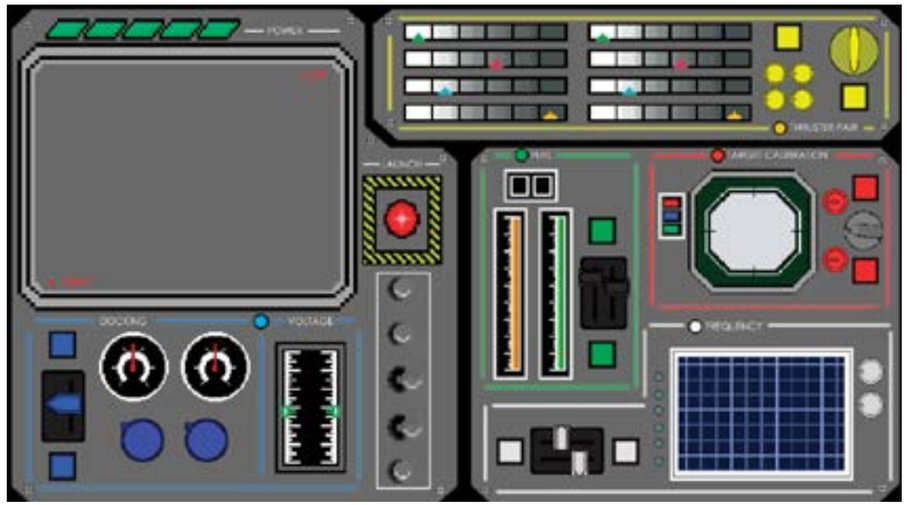


Fig AA6: The final layout

This was just in time, as Sarah and Alex were litterally putting the game together as it was getting finished. You can see on the screenshot on the left that there aren't any actuall game elements (nothing in the white grid, nor in the red radar), because those were dynamically added in at the game's run time so the player could have different win states.

The final sprint

What happened next is something from the movies. Over the course of the next week Alex and Sarah worked tirelessly on getting the game programmed and finished. The implemented the sound effects, added win scenarios, the power module mechanics, the light and success features, every single win and error animation, reloading and randomization implementation, all of the remaining textures and button inputs, and literally every bit of polish and optimization the game ended up with.

It's sad that there aren't any images that we have to share here to summarize the vast amount of insane work that was completed during this sprint of the damned. Sarah and Alex for the entire week literally swapped out on sleeping while the other programmed. There is a record of the chat for those interested, shared by them and myself, and it's literally them back and forth for hundreds of lines figuring out how to implement the game, and occasionally passing out when the other hopped in to start editing.

Throughout the entire week I was in contact with the museum buying as much time as I possibly could. The excuses were fantastic and robust, from things like "Oh I think I have a TV in my warehouse down town, I'll get it tomorrow" and "oh that one didn't work, lemme call by buddy and get a TV from him tomorrow" and "Oh that one is too big for the wall? Ok, gimme a day to come up with a better solution." I'm not proud of these things, but considering how close we were to deploying this, I probably would have done a lot of things if it meant getting our project finished in time.

Were this some form of sitcom or other televised media, the entire week would be summed up with a montage of sorts, where it shows Sara and Alex sleeplessly downing re-bull and monster energy drinks, taking turns loosing consciousness and haphazardly smashing in art assets. There would probably be some kind of Rocky-themed music going on in the background as lines and lines of code were created and re-done and created again. Occasionally there would be a clip of myself in the montage doing some sort of slapstick style comedy at the museum like spilling stuff of the TV to buy us more time. The entire sequence would be pocketed by brief instances where the viewer would be able to see the bags under Alex and Sarah's eyes gradually grow larger until the final day when they each passed out on their respective keyboards, somehow landing on the "submit" button.

The sequence would end with Sarah dragging herself to my door with the final version of the game and control board before passing out in my front lawn and being carried away by her roommates/boyfriend.

Vid 4: The game, actually working

The 'finished' game

A number of fun and additional features were added in during that sprint, some of which neither Alex nor Sarah has any recollection of oddly enough.

They did add in some of the more hallmark features of the game though. They were able to add in a way for the user to see constant input in the form of buttons being pressed and knobs being turned. Using the combination of art assets that Kara, Jacob, and Taylor had created, Sarah made an in-game reflection of every single input on the physical board.

There were a few changes that I was able to convince them to be added in as the game progressed, such as the addition of a voice clip every time you win a minigame, which meant begging Chris to drop everything and find us additional media to use, which he was fine with thank goodness. We also added in the working version of the glow effect around each of the modules, which was made more intense as we didn't know how the lighting in the museum itself would hold up to it.

At some point during all this I was able to build a tiny PC specifically to run as cool as it could in a sealed box all day, and get an operating system on it that would work well with the USB controllers we were using. It was funny, Sarah eventually posted the finished version of the game in our chat but none of us could actually play it, as it requires the physical control board. So we all just opened it and

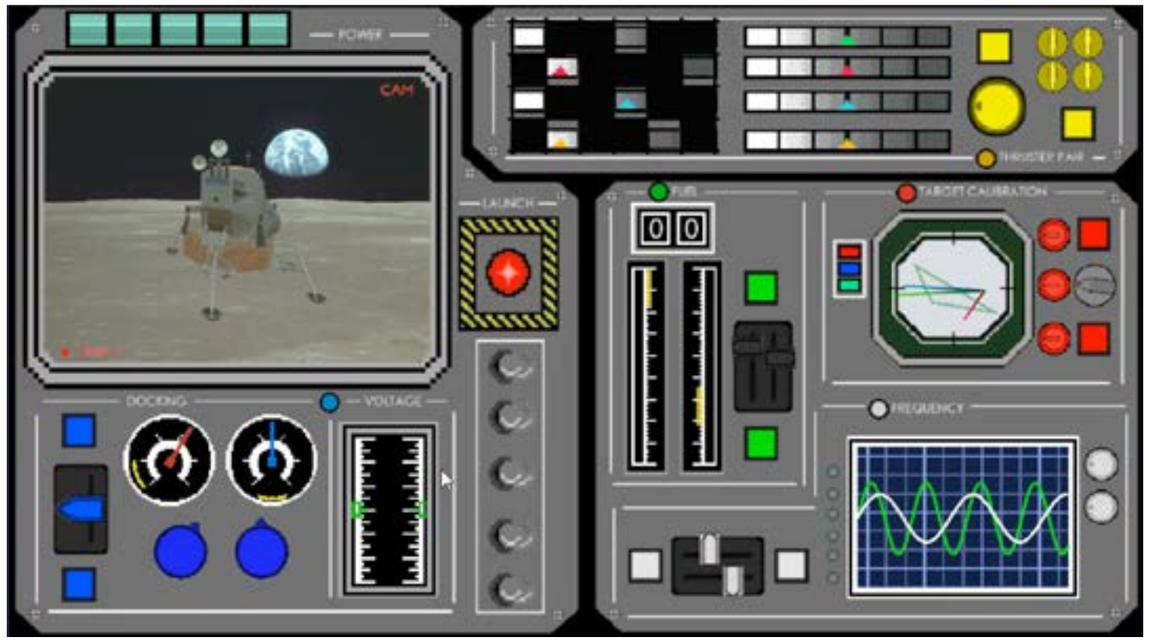


Fig AC: The final game screen

looked the opening screen for a while and said "yep, I hope this works" (Fig AC).

Deployment

The deployment itself was… a bit rough, to say the least. Deployment actually happened so fast we didn't end up having all our stuff together despite our hard work the week prior.

Monday I got a message from Keith that he was sick as a dog, but the podium was ready for pickup. I thought for sure this would be the perfect excuse to extend our deadline a bit, but the museum was ready no matter what. I mentioned to them that Keith was sick, and my car (1997 Toyota Corolla) was vastly incapable of getting the podium to the museum, and they responded saying they would drive to Keith's house with a truck and crew to load it up. That surprised me a great deal, and I responded saying we'd figure out a time that would work.

I was awoken Tuesday morning by an email saying they would be at Keith's workshop in an hour. I frantically called Keith to make sure he was ready to pass it off, and then phoned Sarah, who had the actual game as well as the top panel with the buttons, as she had just finished testing the night before.

I grabbed the computer I had been fiddling with, hoping it would fit in the podium, and sprinted out the door. …only to come back in and realize that we were also the ones providing the TV. I had been using the TV as a stalling point to keep the project going, but hadn't actually done any work into actually obtaining the TV itself. Because I only had so much time before they were expecting to meet us at the museum with the actual product, I just ran into my workshop and took the biggest TV I had been working on. I grabbed any cords that looked even remotely helpful, and booked it. (Fig AD)

Fig AD: There were a lot of TVs to choose from, most of which were very broken.

Because the museum was the capital center historical museum, it was quite literally in the heart of the new capitol building's plaza, which means no parking anywhere close.

Sarah got hilariously lost and ended up parking in front of a nearby bank and just walked up with the front panel and all its wires coming out, which only looked a little bit like a bomb. I just parked several blocks away and ran back and forth with the TV and the computer and the parts. Hindsight says I shouldn't have done that in Florida in August at 98 degrees out, but hey, we didn't pass out so no complaints.

That didn't work at all

So we got lucky that the PC fit into the actual podium just fine, and Keith's case fan also worked like a dream once I wired it up to the motherboard. There were a few minor issues here and there, but I brought my toolkit to get it figured out.

There was a level of comedy to the situation, this was the first time the pc had ever actually run the game, and it was as we were setting it up in front of the client. Needless to say it had some… issues.

Due to it being finals week, Sarah hadn't actually had a chance to test the full game, so this was the first time we were having it all interact together. The major issues were the power module section as well as the dials linking. For some reason with the power modules, the actual values would sometimes overwrite each other. That resulted in not being able to actually win the game due to not being able to unpower the units. In the game, it had been set up so that the player only had a certain amount of power. As soon as you turned on more than two modules, you overloaded the power and the game freaked out. The issue was, as we discovered when putting it in, depending on which two modules you left power on to, the game would be stuck in a permanent mode of overload and have to be reset. (Fig AE)

Fig AE: Hour 3 of Sarah and myself attempting to get it working in the museum

We also hadn't actually gotten the joysticks to synch up with the USB ports, so all of the dials were controlling other random features of the game and it was all over the place. Sarah had her laptop so she was able to start debugging as I got to work adding in a power button and other hardware features for the museum staff to use so they wouldn't have to figure out how to turn it on and off through the code.

Keith had sent a TV stand for the wall, which the museum staff was happy to immediately drill into and hook up with my TV, which I only then noticed happened to be covered in Nutella, because I hadn't cleaned it before throwing it in my car and running it over here, so we quietly took some paper towels from the bathroom and polished it up a bit.

All in all, it was about five hours before we were actually done. Between the hardware issues, the game's glitches, and the fatigue setting in, we just started throwing things out at the end. For the power module issue, we just got rid of the fail state. For the sliders problem, we just removed the functionality of the white sliders. For the launch switch that was having issues, we just removed it from the game completely.

By the time we were done the museum was closed and we were one of the last ones out with the staff. But the game was deployed. It wasn't as functional as we wanted, but it was working. And after three weeks of pure chaos, that's really all we could ask for. (Fig AF)

Fig AF: Never has a more victorious picture been taken. Don't trust our smiles, we were absolutly dead on the inside by this point.

The conclusion

That wasn't the end of the game's creation, not by a long shot. As the weeks progressed we added more to it and fixed it when it broke (or when kids broke it). We were able to add a number of simple features, and discovered that our original design for the game wasn't…. actually good. So though a great deal of reiterations and new versions we would come in to the museum with a USB about once a week and update what we could. In the time it was deployed we more or less ended up changing the very foundtation of the game from the ground up as we saw people play it and adapted it to be more enjoyable for them.

But while all of that was its own adventure, it wasn't on the same crazy time crunch everything had been up to deployment. We always had at least one person on staff to drive down to the museum any time the game did something it shouldn't have, and we eventually got it to a state that was… somewhat playable. But that's really its own story, more focused on design and iteration than the actual deployment.

But all in all, creating a game from scratch, it's own custom hardware, design, art, sound and everything else all made custom in house in the span of a few weeks with a budget of zero dollars and a small team of people who hadn't even worked on something like this before, it came out pretty well. At least, we like to think so.

Directed by…

Lucas von Hollen

~and~

Keith Roberson

Design by…

Bryan Clark

~and~

Jacob Wharton

Programming by…

Sarah Hall

~and~

Ethan Shelton

Art by…

Kara Raya

~and~

Taylor Lundy

Sound by…

Chris Tonner

~and~

Logan Harold

Animations by…

Alex Wood


110A