Intergenerational Gaming in *Kerbal Space Program*

Eric Klopfer, MIT Scheller Teacher Education Program/The Education Arcade
Oren Klopfer, McCall Middle School

Abstract: *Kerbal Space Program* is a detailed, challenging, and engaging simulation game just released in its final 1.0 version. The high fidelity of the simulation along with the breadth and depth of choices make the game interesting and represent a departure from the simplicity of modern mobile simulation games. Other elements of design, including the iterative, turn-based play, and required knowledge make it amenable to intergenerational play by providing roles for the knowledgeable parent and game-skilled child. This Well Played session by a parent and child duo walks through the game play itself and some of the interesting parent-child/child-parent interactions promoted by game play.

What is *Kerbal Space Program*?

The flight simulator was once a top selling game within the PC gaming industry. It has now largely disappeared, with Microsoft finally bidding adieu to the latest iteration of their *Flight Simulator* line in 2013. So why would a flight simulator that is more complex, less scenic and requires you to assemble your own flight vehicle from parts capture the interest of so many Steam Early Access players? We're not sure, but we have some ideas, and we'll explore that in this Well Played session.

The game of interest is *Kerbal Space* Program, a "space program" simulator which recently exited a long Beta on Steam, and is now in version 1.0 (Squad, 2013). *Kerbal Space Program* (*KSP*) is a highly detailed simulation, in which you play the role of a space program director who must build rockets, staff the rockets, and ultimately pilot them with the goal of getting them into orbit, to the "Mun" or beyond. It is also a game that has some ideal elements for intergenerational game play.

The Challenge of Intergenerational Game Play

Most of the popular literature on gaming for parents focuses on warnings about addiction and violence. The first generation of video gamers has grown up (Eric) and now has children of their own (Oren and his sister, Maya). This should provide an opportunity for video games to bring parents and children together, but instead we see a growing divide around this medium.

Some organizations have stepped in to try to fill this gap. The Center for Games and Impact at Arizona State University (Crawford, 2013) has created guides for parents to better understand impactful games that their kids might play, while the Joan Ganz Cooney Center has published some research and more general guidelines (Chiong, 2009) that aim to help parents think about how to play games with their kids. These issues continue to receive popular press (Shapiro, 2013).

While games designed for multiplayer interactions may seem like the obvious choice for such collaborative play (and, truth in advertising, we like to play *World of Warcraft* and other multiplayer games together), single player games may also be the focal point of such interactions. There is an opportunity in such games for cognitive apprenticeship (Brown, Collins & Duguid, 1989), where a parent can model and provide coaching on particular strategies, while the child can similar model other areas of in-game expertise. There is great research supporting the kinds of "in room" interactions between players that can *KSP* exhibits some interesting properties that make it a single player game that is still amenable to collaborative play.

First, the game is inviting. While there is complexity underlying everything that you do in the game, there is also engagement through feedback. While that feedback is occasionally positive, it is often negative. But the feedback loop is tight and engaging, providing the fodder upon which a parent can reflect, and coach their child to do the same.

Second, the game is really complicated. Did we mention that already? The complexity is not arbitrary but rather comes from the detailed physics modeling. The complexity invites participation from multiple participants. But the particulars of the complexity mean that the skills of a knowledgeable parent who might remember something about physics, or even what the rockets that the United States used to launch looked like, can help a child who brings with them proficiency in manipulating the 3D components of the rocket, and some piloting skills. This creates a well-balanced team where the child is both master and apprentice in different roles.

Third, the building stage is iterative and turn-based. There is a lot of opportunity for thought, reflection and input. One player might have the controls to browse through the available parts, while the other player still has many opportunities to critique design, suggest other parts, point to where a part should go, or cite previous data that might inform the current design. The complexity of the space, feedback and lack of time pressure all contribute to this process, as does the balanced failure state mentioned previously. In fact, without this dialog, the game is significantly more challenging. Taking out a new part from the inventory requires explanation and reflection when playing with another player (older or younger). This is an important learning opportunity for both players. The piloting stage is more real time, but still may be slowed down to allow for similar interactions in this stage.

Failure is Sort of Fun

The game takes place on the planet Kerbin, an earth-like planet that is inhabited by Kerbals. It has several different modes including a *Sandbox* mode in which you have access to all of the possible parts and unlimited resources, a *Science* mode in which you are given scientific challenges and limited parts and sensors to accomplish those goals, and a *Career* mode in which successfully completed contracts and missions provide resources to build more complex space vehicles and equipment for increasingly challenging missions.

While many games might encourage players to start in a more bounded mode that provides additional structure, leaving the sandbox for later, *KSP* has such a steep learning curve that such constraints are not necessary in the initial phases. Unlimited resources and parts still inevitably lead to a rocket that at best explodes (Figure 1) shortly after launch.

Figure 1: One of the satisfying explosions in *Kerbal Space Program*.

Failure is a great way to learn, and an often overlooked design feature of games (Juul, 2013). For many good games, learning comes not only with success but with the oft repeated failure. Persisting through such failure requires (and maybe generates) grit, but games can also soften the blow of failure. The explosions in *KSP* themselves are quite satisfying and provide just enough incentive to try again. In addition to the explosion, however, the Kerbal pilot (Figure 2) is lost (temporarily) in such explosions. While it may not seem like a huge impact to lose a comical, easily replaced character, the attachment to the pilot is not insignificant. In many ways this is the key to the success of *KSP*; the failure state is well designed. Failure provides just enough negative feedback to cause the player to rethink their design and piloting, but enough levity that the failure isn't devastating.

Figure 2: A frightened Kerbal pilot.

And that is good, because there is a lot of failure in *KSP*. The game has an extremely steep learning curve. It turns out that building, launching, piloting and navigating a spacecraft *is* rocket science, and *KSP* makes that very clear. An initial challenge might be to build a simple rocket that can fly up a little into the atmosphere, eject the control module with the pilot inside and land safely back on earth. But there are many points of failure in such a plan (Figure 3). Are the fuel tanks heavier than you have lifting power for? Did you remember to put a parachute on top and decouplers between the stages? Are the stages in the correct order? Are the engines the right match for the fuel tanks and sources? Are the aerodynamics sufficient to keep the rocket stable? There are many choices, which make the resulting rockets deeply personal, and equally as challenging to get it right.

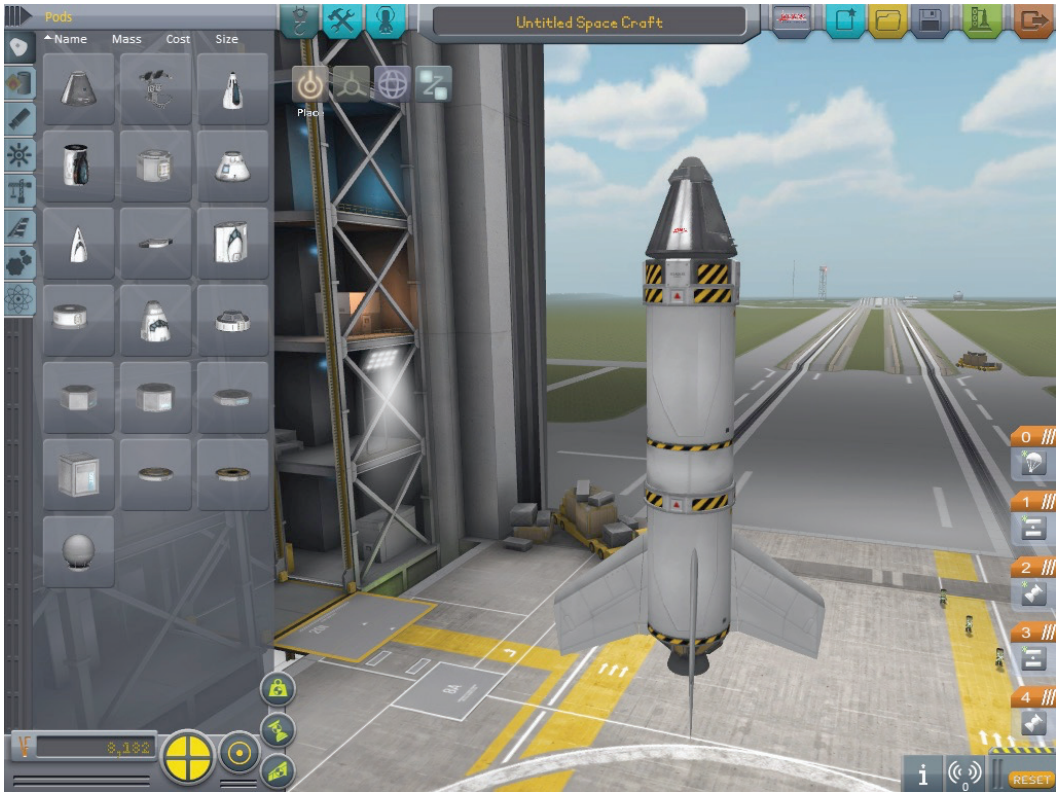

Figure 3: A partially assembled rocket and some of the available parts.

Failure also provides a useful context for learning in *KSP*. Players love to test the boundaries of systems in games–they might try to jump off a cliff, hit their traveling companion, or race to the end ignoring the prompted goals along the way. In *KSP* such tests lead to important learning. Trying to create the largest explosion on the launch pad means figuring out what fuel tanks provide that potential. Getting a rocket to burn up as it leaves the atmosphere also requires a knowledge of fuel tanks, engines, and knowledge of the weak points to build into your rocket. While success may be slightly more challenging than failure, specific failures require building deep knowledge of the component systems in *KSP*.

The Devil is in the Details

Launching a rocket doesn't need to be this challenging. At least launching a simulated rocket doesn't need to be this challenging. But *KSP* has opted for a very detailed and accurate simulation of the physics and engineering (with a few exceptions like multi-body orbiting). Many games, even simulation games, opt for low fidelity simulation. *KSP* has adopted a detailed and accurate physics model, which I'm told (by aeronautics students) is quite lifelike. Getting better at aeronautics (rocket science) seems to make game play easier. This is a desired quality in a game designed for learning (which *KSP* isn't explicitly)–getting better at the underlying content should make one better at the game (and also hopefully vice versa).

The details span the physics, the library of parts, and the community that surrounds *KSP*. One can read up on the different aerodynamic properties, tolerances and propulsion properties of the available components. These are not mere labels, but instead are accounted for (Figure 4) in the simulation. There are supporting tools to help visualize how these properties combine to create a center of mass, thrust and lift.

Figure 4: A visualization of the center of mass, thrust and lift on a rocket (left) and detailed information about the properties which combine to create those (right).

Once the rocket is assembled it needs to be piloted. If the rocket is well built, the challenge getting it off the ground isn't too great. But once it is off the ground, getting it into orbit is fairly challenging. Once again the game does not shy away from accuracy introducing terms like *apoapsis*, *periapsis*, *prograde*, and *retrograde*. These terms may be intimidating, but an experienced player develops a feeling and intuition for what these terms mean, making them less scary if they can get that far.

The game has built in tutorials, but in their current state they are of fairly limited use. There is a very active community that does produce copious materials. There are wikis, tutorial videos, and mods that introduce new parts.

So who would play such a challenging game? Us.

Getting Started

We had played some of the early betas of *KSP*, but as version 1.0 approached we picked up the system again to play with a more evolved system. As a Steam game, it is available for play on many platforms–Mac, Windows and Linux. We built a Steam Machine (a PC running Steam OS) in the TV room to play games on the big screen. This also means that these games are played in a common household space, not solo in a private space.

Launching the near final version on the Steam Machine for the first time we tried to assemble a rocket from components that we could make sense of – a fuel tank, an engine, a command pod (where the pilot sits), a decoupler (to allow the stages to separate). Some of this knowledge came from previous experience and some came from watching rockets launch. Rocket launches are no longer the public display that they once were 40 years ago. So the idea of the multiple stages of rockets was somewhat foreign to the younger of us and required some *coaching*.

But the interface prompts for this design, highlighting the sequence of events that your design will produce (see the lower right corner of Figure 4, where events 0-4 are noted ending with the parachute at stage 4).

The first rocket doesn't take too long to assemble once you find your way around the interface navigating pages of fuel tanks, engines, aerodynamic components and structural components. We built a rocket that looked pretty nice in that it looked like an actual rocket. The older of us sat on the couch and guided construction, while the younger of us had the mouse and keyboard to do the actual work (though we did change up this sequence periodically). We counted down towards ignition and launched the rocket. Somewhat to our surprise it took off. It got to about 10,000 meters before we had exhausted the fuel in the tank and and ejected the command pod which promptly fell back towards the surface with Bill inside. He didn't make it, as we had not sequenced the parachute correctly that time and he crashed into the planet's surface.

We made a few notes to *articulate* our strategies and took some time to *reflect*. First, we ought to resequence the parachute deployment. Second, the command pod didn't go anywhere, so it needed something to power it once the big tank was decoupled. The parachute resequencing was easy. But the tank and engine for the command pod caused some debate. Should it be a small tank and engine? The main tank got us up pretty far and pretty fast. Maybe we just needed a little boost to get out far enough to get into orbit. Or maybe we weren't that far out at all and we needed a much bigger boost. Would a strong engine and a small tank be sufficient? Or would that simply cause a little blip in our trajectory. These decisions required a lot of discussion, which needed evidence to support them. The older of us tried to come up with as much evidence as we could muster.

To make a long story short, the second, third, and fourth launches showed minimal progress (Figure 6) which provided us ample opportunity to reflect. We got Bill, Bob, Jebediah and the recently introduced (and long overdue) Valentina back to the surface safely a few times, but never made it much higher. That is when the younger of us said, let's scrap this and build a really big rocket since this little one wasn't taking us far enough. The older of us could have given a rationale to support why this wouldn't work, but learning by doing is often a more effective way of teaching and parenting. So the younger of us built a great big rocket with massive engines, tanks and wings. Upon launch it didn't go anywhere. Some modifications got it as far as breaking apart on the launch pad and exploding in a spectacular fashion, so we stayed with that for a while.

Learning Through Research

As much as one can learn by doing, there also comes a time when that doing can be supported by Just In Time research. We knew that we should go to the Internet to find some resources that might help point us in the right direction. An interesting thing happened at this point. As is often the case while playing *KSP* in the TV Room, we were accompanied by Maya, who is in third grade. Maya immediately jumped to YouTube to get video tutorials on game play as she has often drawn upon before (copious resources for Minecraft come in this form). But the younger of us (also Maya's brother) felt this wasn't the right medium for getting the information that we needed. We needed to be able to scan through information, read about components, and tailor the information to our own needs. Videos (in Oren's words) just give you an answer without an explanation. Textual and graphical tutorials would be much more useful in this case. Maybe Maya learned that as well.

This is a great moment in any 21st century parent's life when they realize that their child has developed some fundamental media literacy skills. Indeed we found some textual tutorials that seemed reliable (on a *KSP* wiki) quite quickly and were able to use that information more easily than if we had to watch a whole video (many of which detailed all of the person's failures before success, or simply documented the success without any detail on how we could do the same thing). We learned some important things – turn early (don't wait until you are out of the atmosphere), use multiple stages to get rid of the weight of the tanks after they are done, shoot for about 70,000 meters, which is the low end of orbit, and to get into orbit you need to accelerate prograde as you near the apoapsis. These terms required some research, which the older of us was able to do and reduce to common terms. While this didn't get us into orbit it got us very close.

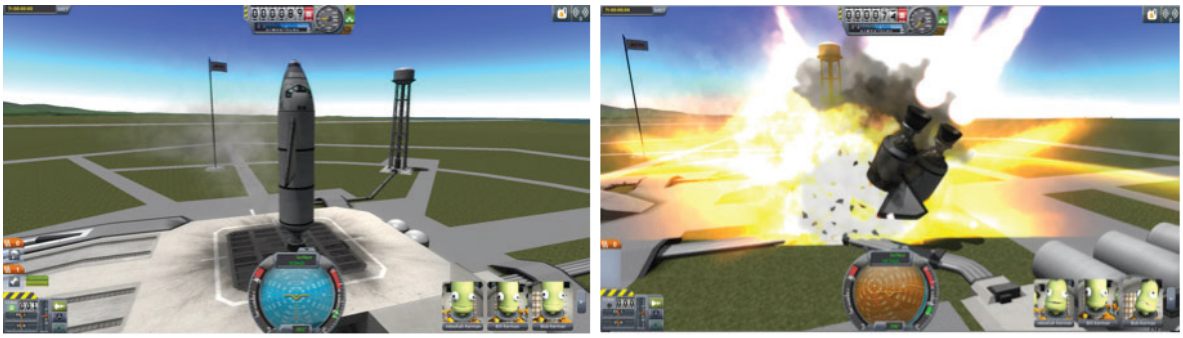

Figure 5. Some of our failed attempts in *KSP* where rockets were too heavy (left) or not stable enough (right).

Approaching Orbit

For several days we were stuck. We had the basic concepts down, but always seemed to run out of fuel as we approached the apoapsis (peak orbit). This caused us to scrap our designs several times. We played nearly each night and would exchange ideas over breakfast or dinner, and came in with ideas for different designs. How about if we had a giant first stage? We played with that idea for a while. While the older of us was very goal oriented and wanted continual progress, the younger of us was gratified by side goals that we just invented. Could we get the giant rocket to go straight down and crash into the launch pad? Yes. Maya tracked the death and survival of our four Kerbals over the first few days. We lost a lot of them. But some survived.

We did get the giant rocket to go straight up fairly far. And then we noticed something strange. When it got far enough out it exploded. What was causing this explosion? Dialog really helped here. Did it crash into something? Let's watch closely and look at the map view to see if there is anything that it could crash into. No, it didn't crash into the Mun. Did it burn up? While it got really hot at one point, the place where it exploded was far from there. That shouldn't be the cause. Was it air pressure? The older of us helped devise a series of tests and we went through a list of tests and still couldn't find the cause. Then the younger of us got a bonus afternoon session (time to *explore*) and tested some more possible causes. It turned out it was the time warp. You can accelerate time in the game, since space travel can take a long time. If you accelerate time, the ship explodes. But if you put time back at the normal pace at about the place it typically explodes, the ship doesn't explode. The older of us attempted to explain how something like that could happen by saying that this "acceleration" actual skips some steps to make it so fast, and that in turn introduces error which can cause these kinds of things.

With that solved we went back to design, and importantly some additional research that the younger of us continued to do. That informed our design and we made some changes that had more to do with piloting than construction. But this piloting required additional steps. We took turns piloting and reading out the sequences of when to turn, how much to turn, when to use full throttle and when to throttle down. We had a lot of debate about the right speed to hit later in the launch. Would faster get us there faster? It might, but it will burn fuel that much faster. And going too fast introduces friction (which we sometimes saw as the ship nearly burned up) that we want to minimize to use as little fuel as possible. Eventually we found success.

Figure 5. A successful orbit in *KSP* is shown looping entirely around the planet.

This marked an important milestone and of course posed the next question – how do we get our Kerbals back? This process of design, build, and test (along with research in various forms) is a great way to interact and even allows for differential time spent on the game while both of us still feel a sense of progress and ownership.

Who We Are

Eric Klopfer is a professor and designer of educational games, with a background in simulations. He has researched and developed a variety of Science, Technology, Engineering and Math games, and sees *KSP* both as a way of bringing interesting detailed simulation games into formal and informal learning environments, and as a way of bringing legitimate adult and child roles into games. Oren Klopfer is a rising seventh grader who is a game player and also likes to dabble in game design. The duo has spoken together previously on a panel at PAX East. We will recreate what it is like to get started in *KSP* with the fun of failure, success and collaboration.

References

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18* (1), 32-42.

Chiong, C. (2009). Can Video Games Promote Intergenerational Play & literacy Learning? Joan Ganz Cooney Center. Retrieved from http://www.joanganzcooneycenter.org/wp-content/uploads/2010/03/intergen_final_021210.pdf

Crawford, J. (2013). Move over, Monopoly: ASU researchers find families bond over video game play. ASU News. Retrieved from https://asunews.asu.edu/20130709_families_videogames.

Juul, J., (2013), The Art of Failure, Cambridge, Mass.: MIT Press.

Shapiro, J. (2013). Research Says Parents And Kids Should Play Video Games Together. Forbes. Retrieved from http://www.forbes.com/sites/jordanshapiro/2013/12/04/research-says-parents-and-kids-should-play-video-games-together/

Squad. (2013). *Kerbal Space Program*, Steam Early Access edition. [PC video game]. Mexico City: Squad.

Stevens, R. Satwicz, T. and McMarthy, L. (2008) In Game, In Room, In World: Reconnecting Video Game Play to the Rest of Kids' Lives. In The Ecology of Games, Salen, K (ed). Cambridge: MIT Press.

Acknowledgments

We'd like to thank Maya Klopfer for her faithful tracking of the fate of our Kerbals and also for some helpful suggestions in the process.