Turbek.com

A few rough notes about the development of the Invisible Maze game

Stephen Turbek23 December 2013

I am no serious programmer, but I look to try out the odd project and have some fun. Here are some notes on the development of Invisible Maze, a free HTML Canvas / JavaScript game for touch devices (Targeting iOS). It works in Safari / Chrome on the desktop, using the arrow keys to steer.

This is how my 8 year-old son explains the game. "Well, it's about this little guy named Mr. Wiggles in a maze and the walls appear and disappear so you got to remember where they are and when you hit a wall it never disappears and there's bouncy walls that bounce you in the other direction.'

The boys are very interested in games, so we had an idea to make a game as a way to for them to learn about programming and me to learn HTML5 Canvas JavaScript programing.

Designing and building a game with 2 young boys takes a lot of patience, but was a lot of fun. They really loved the idea and would help sketching, and of course endless sessions of beta testing! Their measure of success was very clear: get something in the App Store to show their friends.

Inspiration

One of the inspirations was a game I played as a kid called Pathfinder The basic idea was a two person player game where you created a maze with colored clear plastic chips on a grid. You each created a maze that was hidden from the other player, battleship-style, and then you try to find your way to the maze before the other person. At least that's the way that I remember the game. I vaguely remember that the game was too hard to be fun.

Another inspiration was the photos game labs game X-type. It's an action-packed shoot 'em up game with explosions another great effects that showed what you could do with Canvas and JavaScript in the iPhone browser. The game is also originally inspired by the game Dots, which had a very beautiful drop animation is very clean and design-y.

Video games are today's gesamtkunstwerk. They contain a variety of crafts: audio, visual, interaction design. To do it well takes a team and much work, way more work than one dad can muster. I remember hearing that Angry Birds was Rovio's 57th game. The App Store videogame market has really matured; top-tier games have incredible production values. Some games, like 'dumb ways to die' even have advertising campaigns.

Evolution of the game

Like any personal project, the hard part is slowly developing the idea with the time you have. But it turns out there is a lot of time in between responsibilities to daydream you picking up the kids or washing dishes. Once the basic gameplay was established, then you could have fun with little side projects like adding custom fonts and sound.

The game went through a series of evolutionary steps. The first step was developing the idea of basic concept of multiplayer to single player. This was done because multiplayer was just too hard we always meant to get back to it.

One key insight was that the basic interaction gesture had to be fun to play. You can add all kinds of gameplay elements like other characters and special bonus points and animations but the essential interaction, the Zen Archery of interaction had to be a pleasure in itself. In Angry Birds, the pulling back of that slingshot and the release is such a joyful motion that the rest of the game can be built around it.

Years ago, I made a very simple Shockwave game (remember those?) where you could shoot flies. There was no score or levels, just an endless number of flies coming slowly into the window to be shot. It could have been boring, but some people found it relaxing. The key was that the flies were annoying and killing them was satisfying. The shocking contrast between the bumbling, buzzing flies and the gun sound effect made the game.

However, early iterations of the game was released sterile and cold. Simply changing the character from a circle to a face shape did a lot. The cockeyed smile was actually a mistake, as I tried to learn how to draw using Canvas. Part of design is knowing when to keep your mistakes.

Likewise early user tests show that the game had some serious flaws. The most obvious which was that people thought that you could go through the walls when they were invisible. As an attempt to address this problem the game was renamed from PathFighter to 'Invisible Walls'.

People still struggled with this concept, having been taught by every other video game, that when a door disappears, you can go through it. There was an instructions screen, but people were not going to it or reading them.

The basic concepts and gameplay needed to be established immediately and effortlessly. The solution I came up with was to play a little animation of the game with some contextual information explaining what was happening only then was the start game button displayed other you can touch it any time.

The home screen was in fact a maze that one could, in theory, play. Reusing the gameplay animations meant hacking together a fake game which, of course, led to some really messy code. Someone should do something about that someday.

Giving the game personality

For a game to be a huge hit it's more than just the gameplay. People have to personally respond to the character. NOM NOM the main character of "Cut the Rope" is a great example. In addition to giving Mr. Wiggles a personality and a name we also had to make it clear that it was bad when you hit a wall. The answer was to make a sad little cross eyed face that exploded when you touch the wall. The pie-slice disappearing Mr. Pac-Man was the inspiration.

Adding a custom font cannot be overstated as a way to give the game personality.

Audio design is one of the subtlest aspects. A friend showed how adding audio cues, sweeteners, and background sounds made a categorical difference, yet you could hardly point to a specific sound. Having a catchy tune never hurts. Not having musical talent myself, I took a shot at adapting one of the oldest tunes: chopsticks.

There's a great book called game feel about how the game should respond to a person's touch. As I was taught many years ago in animation class, it's all about the accelerations and decelerations that make a professional animation.

Subtleties like, should it be the length of the finger swipe that determines the speed or the actual speed of the finger (which is much harder to calculate)? Should multiple swipes in the same direction add up, and if so, linearly or exponentially? Linearly gives a greater feeling of control, but exponentially makes the game more unpredictable and fun. As a game designer friend said 'The user has to feel that dying was a result of their action, otherwise the game is no fun.' The same could be said for life.

The consequences of Mr. Wiggles dying also would shape the game. Should it be a fast game where it's not significant to die or should lives be limited and very important? Casual games seem to favor the first, so we left

out the idea of lives and made it a simple timed game. Death simply slows you down, but gives you the benefit of making the wall visible from then on.

Resources

I had heard that sound was not possible in the mobile browser, but there was yet another group, Howlerjs.com, that built a wonderful plug-in for arcade sounds management, like a rapid fire repeating noises and sounds that loop behind other sounds.

There are number of JavaScript-based touch and swipe controllers. http://labs.rampinteractive.co.uk/touchSwipe was simple and seemed to have the fastest reaction. But it lacks the full range of gestures like pinch and zoom.

FontSquirrel.com was a good resource to buy and convert fonts for web use.

Development Notes

I've done a bit of coding before you (look at other projects on the site) but every time I make something, especially with a strong design element, it really brings a lot of empathy for the developers I work with. A really simple idea can be incredibly complicated to build.

A big challenge for this project was discovering some of the nuances of JavaScript development in the mobile browsers. One of the worst even situations I had was when sounds suddenly stopped working in the middle of a rebuild. It worked on the desktop, it worked on the iPhone web clipping (when you save a webpage to the home screen) but didn't work in the iOS browser itself. It turns out all I had done is move the sound command into another function to abstract it, like good to tell me practice. Several hours of mild panic later it turns out that there is an undocumented feature in the Safari browser which prevents sounds from playing until after a user interacts with the page. Putting the same sound function too 'far' away from the touch event action handler had made all sound be ignored by the browser. I did finally fix it, but it almost took me rewriting the entire application. Late at night on a weekend, this is the kind of thing that causes you to question why you're doing this in the first place. It sure doesn’t feel like fun!

I tried several times to learn from better smarter developers myself but it's surprisingly hard to review code with someone. The code was just too complicated to answer architectural questions. For a novice developer like myself concepts like MVC are too abstract, unfortunately, in a part-time project. It is hard to see the benefit of architectural perfection versus adding a new feature. With full-time development,especially something you going to live with for a long time, the answer would be very different.

Development Environment

The development environment was entirely done in Bare Bones software TextMate, and of course, in the browsers. Chrome's development tools are great but Safari's aren't bad either, and I was targeting iOS.

It would be great to have some kind of version control system, but if you're only working by yourself that's a bit more complexity and perhaps it means I developed a personal version control system called Nikki lots and lots of copies each version of the software is saved in number directory so the 61 6263 and there's an index.php the chooses the latest one delivers it to the viewer. The PHP script also minifies and compresses the JavaScript.

By putting the word beta in the directory name it is hidden unless you add ?beta as the query string parameter.This is a slightly obscure way of managing version control but works pretty well for a single developer. The single most important thing for me as a developing developer is to have many, many versions so that I can go back in and see the version when some specific edge case stopped working. It would be great to combine this with the Apple's built-in Time Machine version control. Really what I want is infinite levels of undo.

Obviously if you're working with other people and you have solid test harness, a lot of these problems go away (and are replaced by other problems ;)

The End?

With any personal project it's hard to know when to end. "Invisible Maze" has achieved the basic goals of making a game that is fun to play, but I don't think that's this is the one to quit my job for.

Next steps with the game are a bit unclear. It seems like a fun game to play but it's clearly not a full-featured game. Additional ideas include obviously saving the scores and compare them against other people. Social sharing features. And of course any new gameplay elements such as exploding walls, walls that are buttons to open up other walls, bad characters, and little bonus stars you can collect along the way.

I still have the code for the"Creative mode" the name, of course, taken from Minecraft. This allows anyone to create their own mazes. There was an idea that the player could unlock the ability to create mazes and send them to their friends.

Taking this game all the way would require a lot more dedication and some serious efforts of marketing. Any advice would be greatly welcomed!