2D Prototype Postmortem

Steven Hoover, Patrick Ryan

Not much complexity to this prototype, get to know the Unity 2d engine by recreating the first level of Mario.

Goals

- Figure out Unity's sprite handler
- Get a working camera and character controller include player tracking, camera shake on command, and smooth camera movement
- Get to know how Unity's 2D physics work

Though this prototype was mostly successful, not all these goals where achieved.

What Went Right

A good amount of time was saved by using existing scripts and repurposing them. Between standard assets provided by Unity and helpful people on Google our second goal took little time. For our third goal, we went straight for some tutorials to help guide our experimentation. When spent enough time to learn what we needed when started on the next part.

What Went Wrong

Our first goal is what tripped us up. We stepped into unfamiliar territory with a few assumptions and didn't properly research the subject until time was too low.

Next Time

I believe that we had spent a little more time in the beginning researching all of goals, we could spotted the area that would take the most time and focused more time there.

Conclusion

Through this prototype was not a complete success, it was more a success than a failure. We completed two of our major goals and the one we didn't complete; we found ways to simulate what wanted. Not the best method, but we got a game working. It is also worth mentioning that recreating Mario was more vision than a goal. We knew that Mario was out scope, but wanted to prove that with time we could do it, and I believe we did.

Dragon Hunting Prototype Postmortem

Jason Ege, Shaquille Turner

The dragon hunting prototype was a first person shooter where the player was supposed to shoot down multiple flying enemies with different flying patterns.

Goals

- Make a 1st person shooter
- shoot down the enemies and gain points
- enemies have different flying patterns

What Went Right

We created a shooter that every time you shoot down enemies you gained a certain amount of points and the enemies would disappear. That each enemy you fought had a certain speed and flying style that made game play more interesting.

What Went Wrong

Because of the short time we had we didn't have time to adjust the speed of each enemy and the speed of the gun which made the game somewhat more difficult then we initially suspected.

Next time

I would want to create more detail monsters that could actually attack you. Have different levels to make it more difficult for the player. Would made the creatures have different flight animations, and attack animations.

Conclusion

When we made this prototype we were thinking of the original duck hunter but how to implement with a different style and different game play. I believe we achieve the goal of making a game that is pretty similar but at the same time has different aspects that will make the game a lot more fun if we put more fine tuning.

Ultra Dodge Ball Postmortem

Patrick Ryan, Jason Ege, Justin Loudermilk

A prototype based on the classic Super Dodge Ball arcade game with the twist that rather than knocking out the opponent's players, the goal is to destroy the opponent's play field causing them to fall through the floor and lose.

Goals

- Create a fully functioning two player local game with Unity and PlayMaker
- Build the game to use Xbox controllers for input rather than mouse and keyboard
- Create logic for the ball to destroy tiles only if the opponent does not deflect the ball before it touches the floor
- To create a system to return the ball to the play field if the ball falls through a gap made in the field

What Went Right

We were able to create a fun and engaging prototype quickly with PlayMaker and gained an understanding of Finite State Machines and how to wire them into complex systems. We were also able to program all the logic needed for the game to function through PlayMaker alone with minimal scripting. The feedback from play testing was positive and we noticed that the pacing of the prototype and quick learning curve allowed for short, but highly replayable games.

What Went Wrong

The limitations of PlayMaker not being able to handle arrays required Justin to step in and write a script to handle respawn on the ball after falling through the field. The best PlayMaker was able to do was respawn it at a random location, but Justin's script allowed us to drop it on a random square not already destroyed.

Next Time

The goals for continued development would be to replace the place holder art with finished characters and graphic effects on the ball to enhance the impact of the game. Additionally, adding power ups or trick throws would add to the feel of the game.

Conclusion

This prototype really allowed us to see how quickly a prototype can get up and running using Finite State Machines and PlayMaker. We were also able to learn how easy it is to capture input from Xbox controllers to give us options for input for future games. Lastly, we were able to explore how a simple and quick game can lead to great player interaction and replay. A game does not have to be complex or filled with story to engage and entertain.

X-Ray Prototype Postmortem

Jason Ege, Tyler Niemi, Vinessa Mayer

This game was an interesting learning experience. Not only does it demonstrate color interpolation between two textures to create translucency, but it also demonstrates culling layers using multiple cameras within Unity.

Goals

- Use multiple render layers per camera
- Allow objects behind other objects to be seen with a key toggle

What Went Right and Wrong

One thing that we did right was enabling multiple layers per camera, but we think using multiple cameras in the same game world was a mistake. In the future, we may not need a new camera to display each layer. We can simply change the properties of a single camera, making it easier to manage cameras so we don't have to change settings for each camera for every change. In a bigger game, if we were to apply other scripts and properties to cameras, it would be a lot easier to have just one camera instead of many. For example, if we had five cameras and we needed to change a public variable, we would need to select each camera and modify the variable five times. If we had one camera, we would just need to modify the variable once, saving time and minimizing complexity.

Next Time

If we did this over again, we would start using culling layers instead of trying to make things transparent using exclusively interpolation which is what was tried in the beginning.

Race Car Prototype Postmortem

Jason Ege, Tyler Niemi

The race car prototype was to test the third person camera and movement through the track.

Goals

- Keep the car on the track
- Test a third person camera

What Went Right

The race track was boxed in with high walls so the player can not drive off the track. The game seems to have a good flow with speed.

What Went Wrong

The biggest problem was the camera clipping through the walls of the track. While we looked into solutions, this problem was not fixed before the end of the prototype sprint.

Next Time

If we had more time, we would fix the flipping issue and work on visual animation that would show the car turning.

Rag Doll Prototype Postmortem

Deven Smith, Patrick Ryan

A short prototype to learn how to rig models up for Rag Doll physics and how to apply physics to a rag doll to make impacts seem more realistic.

Goals

- Rig a model for rag doll physics
- Apply forces to a rag doll in direction of attack
- Rig a non human model for rag doll physics
- Ability to shoot limbs and have realistic responses

What Went Right

We learnt how to use Unity's rag doll tool. Which allows for quick and easy set up of human rag dolls. Along with that we saw how a rag doll could be built without the tool to allow for rag doll physics on non human objects. Also we managed to apply force to the rag doll to make it react accordingly to an attack.

What Went Wrong

The short time frame of the prototype did not allow us to implement a non human rag doll. Also instead of using a hit box like we did we should have put a mesh collider on the melee weapon so that we could find the exact point of impact on the object and apply the force in that direction.

Next Time

We would like to implement a non human rag doll to apply forces to. Also instead of just applying forces to the root of the rag doll which on the human is typically the hips, we would like to try having the ability to apply the force to a rag dolls limbs and have the body react accordingly. Next time instead of using a hit box in front of the player we will use a mesh collider or at least a small cube collider on the weapon to record the exact location at which a hit was landed on the enemy.

Conclusion

We went into this prototype hoping to learn how to use the unity rag doll tool and how to apply physics to a rag doll. After learning the tool we learnt how a rag doll is set up allowing us to potentially free ourselves from using the unity rag doll tool. A rag doll is typically set up with a root such as the hips or torso which will have a flat collider such as a box so that the body will typically land face up or face down when knocked down, where the rest of the rag doll body parts with have capsule colliders to allow limbs to roll. Also we learnt that rag dolls typically have forces applied to their root since everything else is a child of the root on the rag doll. This allows you to apply one force to the rag doll to send everything moving.

Escape Postmortem

Vinessa Mayer, Steven Hoover

What we did

The idea of this prototype was to explore the mechanics of being detected by lights and being able to control that detection through hiding, moving slowly and sneaking. To do this, we built a ravine, set up some randomly moving spotlights and placed obstacles that would force the character to have to travel around them and not be in the line of the light.

What worked

Surprisingly, the lights would never see you if you were close but not in the light, or the light was over you, but it was obscured by a rock. Being able to hide and observe the patterns that the lights made worked well and was great for planning.

What didn't work

It was very hard to keep track of where the lights were because of the camera angle and light placing. This led to players getting surprised when a light shown on them from behind. Also, if a light saw you, then it was over. There was no way to lose the light or get away to safety before dying, which took some fun away from the experience.

What we would do different

I believe this concept would be very fun with some tweaks to the mechanics. Experimenting in ways to make the lights easier, but not too easy to track would be a vast improvement, also making most lights simply get suspicious of you if you just brushed them, instead of instantly going into attack and kill mode would allow the player more ways to be tactical. Also, Hiding may be further refined.