Intel® Level Up Winner *Synchrom* Combines Color, Music and Touch to Addictive Effect

One way to take advantage of the growing popularity of gaming on powerful 2 in 1 devices is to combine the best elements of well-known genres into one mesmerizing title. Winner of Best Action Game in the 2015 Intel® Level Up Contest, the intriguing *Synchrom* game has the addictive thrill of classic shooters and puzzle games, then tosses in a layer of pulsing music and stirring colors. It's topped off with a touch-screen interface perfect for today's market, and is easy to extend to other platforms. Mixing depth and playability with an immersive interface, *Synchrom* is doing a lot of things right, and it's worth a deeper dive.

In contrast to how other contest winners approached their projects, Morphiks Studios (based in Rennes, France) did not select a common engine such as Unity* or Unreal*. Instead, they coded their game in HTML5 and used multiple JavaScript* libraries to complete their groundbreaking title. That unique foundation makes their impressive results very interesting for a growing subset of developers.

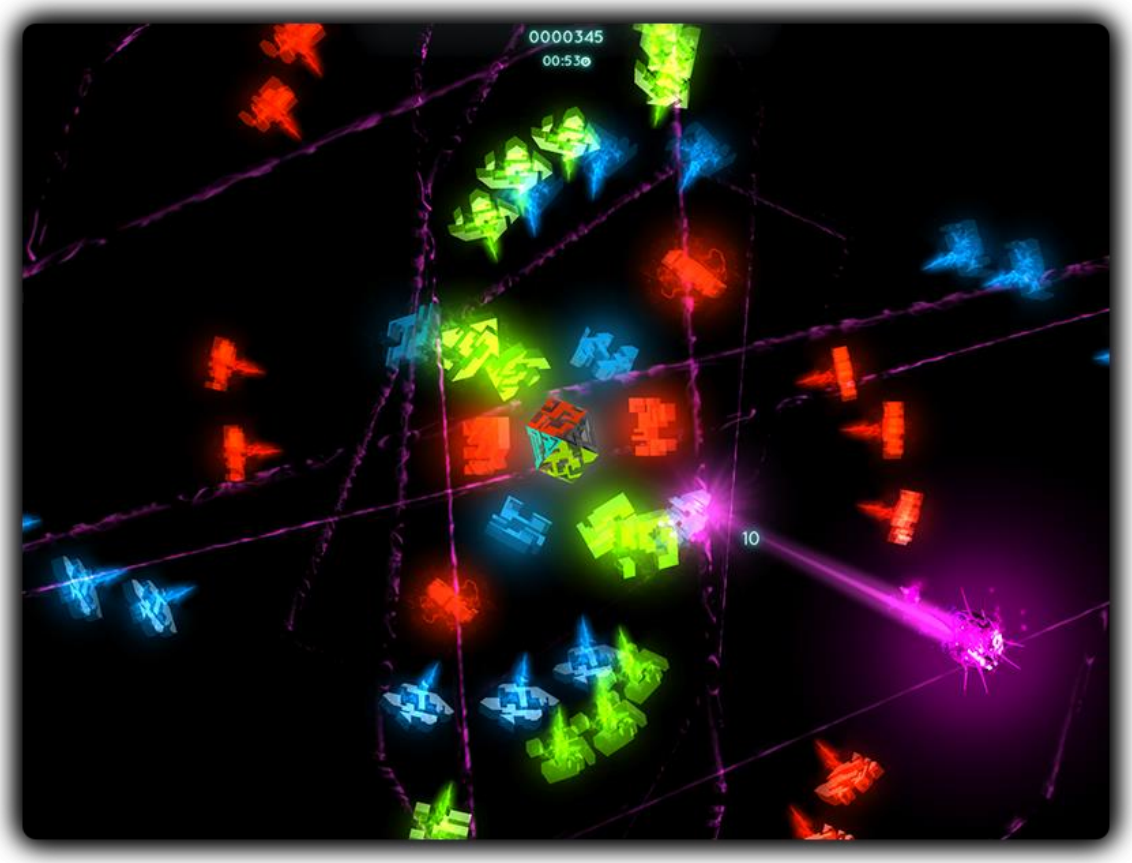

Figure 1. Synchrom is based on rhythmic music, vivid color, precise gameplay and a unique backstory.

Building a New Universe from Scratch

Synchrom is a game with a simple interaction, where players move their fingers or a cursor on the screen in rhythm with the game's music. The goal is to match the right colors whenever the Synchrom device you control fires on the spinning Cryptangle in the middle of the screen. It's clever, it's new, and it takes perfect advantage of 2 in 1 touch technology.

The two principals at Morphiks go by the code names of Arch and Fox; Arch is the engineer, and Fox is the designer. While Arch favored RPGs and city-building games as a kid, Fox could design a completely new tabletop game on the spot–out of paper and rocks if necessary!

When they first got started, Arch and Fox wanted to work on a project that just the two of them could complete without getting overloaded; and, from a practical business side, they wanted to get to market quickly. "We listed the things we liked—including music and being in the flow—and we started to design this game," Arch said. "We came up with a physical interaction that required staying highly focused on a single point on the screen, while at the same time being acutely aware of what is going on around that point. That was how we started to develop the user interaction."

They then created an intriguing backstory for their game, populating a new universe with a race of machine gods called Cryptangles, which originate in another dimension. In the center of this new universe is the mysterious Synchrom: a device controlled by the game player that channels pulses of light energy to fight off the Cryptangles.

Morphiks got involved with the Intel contest because it made so much sense given where they were with their project. They hadn't entered any contests yet, but the game was far enough along that the deadline looked doable. They were already drilling down on what was possible with a 2 in 1, and because the contest emphasized new touch-based technologies, they plunged right in. They wrote their first online demo in HTML5 and quickly put the game out for feedback.

In one instance they got user input by throwing a party in a local gaming bar that had just opened. "Seeing the game on the big screen, with crowds watching the action, was a real treat," Arch said. More, bigger festivals followed; and the bigger the screens, the higher the enthusiasm. "We had four hundred different people play the game by the end of one weekend. It was huge." By the time the winners for the 2015 Intel® Level Up contest were announced, *Synchrom* was on the list.

Using HTML5 Provides Development Challenges

Synchrom started from the ground up, no game engine to rely on. To support attractive 3D graphics in HTML5, Arch and Fox incorporated JavaScript libraries and APIs such as WebGL. Cascading Style Sheets (CSS) supplied the user interface. Those technologies were the basic building materials, but they were only the start of construction. The major challenges are described below.

Working with CSS

Using CSS to form the user interface created many performance issues on smartphones and tablets that the team overcame. "CSS really consumes resources," Arch said. "You have to apply CSS very well, otherwise the action will lag a lot." One workaround is to avoid using too much scaling on CSS actions. Any transformations or any movement on the CSS side costs a lot of bandwidth, so Morphiks was very careful about using them. Arch and Fox had to balance the positives CSS brings to the interface against the negatives of resource constraints.

One trick was to continually remove the CSS overlay and animations for the UI and keep them to a minimum, because otherwise the workload would freeze the main thread upon rendering. CSS renders only the text; everything else is sprite-based. Morphiks advises to make everything a sprite and avoid CSS whenever possible.

Overcoming Memory Issues

Arch said that using HTML5 and JavaScript libraries brought some unique complexities. "JavaScript has a weird way of using memory," Arch said, "so you struggle to build an efficient game with a good frame rate. That part was difficult."

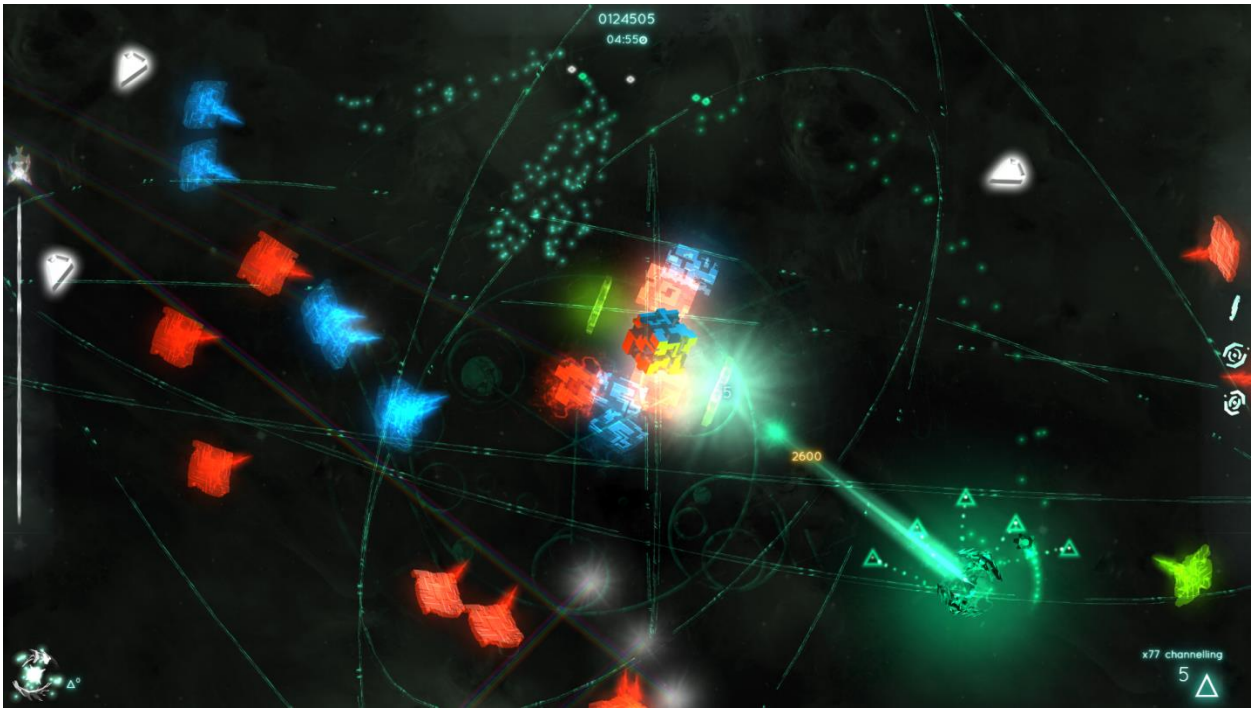

Figure 2. Objects such as particles, sprites, gems and Cryptangles shots are not disposed of when they disappear; instead, Morphiks stored them back into an object pool to reuse them. This is the best way to avoid memory leaks, heavy JavaScript garbage collection, and instantiation costs.

Customizing Animations

Animations were also a huge challenge, because *Synchrom* uses geometric functions, not assets. Arch learned to share their functions into buffers to reduce memory and draw calls. "I made most of the animations in a programmatic manner," Arch said. "We are not using Blender*, the rendering software. Our animations are all made with numbers." In other words, Arch had to enter the base of the rotations and the base of the movements manually, as equations, rather than use an asset library. By tweaking an equation one number at a time and observing each result, they learned which change caused what effect. They would continually tinker away and observe what happened. While this was painstakingly slow, it did create interesting animations in the end. "But it's really not advisable to do it this way," Arch said.

Monitoring Collisions

Because the game is in 3D, Arch worked hard on the concept of dimensions. The Cryptangles are essentially a 3D model. Cryptangles suddenly appear in scenes and arrive at different places, yet the interaction is closer to a 2D effect. "We had to be inventive with collisions," Arch said, because they had to be able to detect hits within a few milliseconds. The system needed to know if the shot touched the "good" color, which was a key point of the game. If this failed to work right, the game wasn't going to be playable as they envisioned it. Because they had no processor budget for mainstream physics engines, they turned to three.js, a 3D library that helps developers manage 3D scenes and objects, to provide code to help with simple collisions.

The following code samples show how Morphiks managed collisions with the accuracy they needed.

First is the algorithm for collisions:

```
     // Here is an optimized algorithm we use for shot collisions.
     // Uses Three.js library for the math, but the principle is the
     // same with any engine

          _colliders = /* Object selected for collision, do not collide
the whole scene - only what's needed*/

      // Create the 2D/3D RayCaster
            var rayCaster = new G.RayCaster2D3D(
                player.getPosition().clone(), // Ray Start aka player
position
                this.boss.position.clone().sub( player.getPosition()
).normalize(), //Ray directed to the center of the cryptangle
                _colliders,
                player.distanceToBoss //length of the ray; you don't
need to RayCast elements that are behind the player or above the
Cryptangle
            );

            var intersects = rayCaster.collide(); // Ray Cast Bounding
spheres only and sort them

            var intersect = rayCaster.crossCollide( intersects,
this.camera.position.z ); //Ray March spheres from the camera on the
real geometry until you find a collision point

         // intersect now stores the value of the nearest point to the
player visible by the camera very accurately
```

Next is the class for collisions:

```
/**
* Creates a RayCaster for 2D/3D collision
*/
G.RayCaster2D3D = function( origin, direction, objects, far ) {

            this.origin = origin;
            this.direction = direction;
            this.objects = objects;
            this.far = far
        };

     /** Collision sorting function */
        G.RayCaster2D3D._descSort = function( a, b ) {

            return a.distance - b.distance;
```

```
        };
```

Then they could manage the collision between the player action and the Cryptangle:

```
        /**Collide elements from the player to the center of the
Cryptangle*/
        G.RayCaster2D3D.prototype.collide = function() {
            var intersects = [],
                object,
                raycaster = new THREE.Raycaster( this.origin,
this.direction, 0, this.far ),
                position,
                intersect;

            var sphere = new THREE.Sphere();

            for ( var i = this.objects.length - 1; i >= 0; i-- ) {
                object = this.objects[ i ];
                if ( !object.geometry.boundingSphere ) {
                    object.geometry.computeBoundingSphere();
                }

                raycaster.ray.origin.setZ( 0 );
                raycaster.ray.direction.setZ( 0 );

                sphere.copy( object.geometry.boundingSphere );
                object.updateMatrixWorld(true);
                sphere.applyMatrix4( object.matrixWorld );
                sphere.center.setZ( 0 );

                intersect = this.intersectSphere( sphere,
raycaster.ray, object );
                if ( intersect ) {
                    intersects.push( intersect );
                }

            }
            intersects.sort( G.RayCaster2D3D._descSort );

            return intersects
        };
```

Incorporating Audio

Arch said he struggled with connecting the game to the music. He needed complete precision when the Synchrom fires, and when the animation and sounds trigger, because the player has to hit what they aim for and earn rewards for doing it. Using regular HTML5 for the action did not work, so the team tried a JavaScript library called Howler.js for effects, incorporating a new API called Web Audio API, which

brings native sound capabilities to HTML5 in a predictable way. That worked. “If not for the Web Audio API, we wouldn't have been able to make the game sound the way it does,” Arch said.

Creating the Executable

Making an application out of HTML5 and JavaScript libraries is still not common, but it is gaining traction. Moving from a Web app to a game that’s playable without a network connection requires even more tools, which the team quickly tracked down. NW.js enabled Arch to call all Node.js modules directly from the game’s JavaScript files.

Figure 3. Online leaderboards are tricky to maintain when the balance of the game evolves in patches. Morphiks reset the leaderboard when they released their October 2015 patch.

Preparing for Distribution

Next, they needed to integrate with the Steam API, because Steam is a leading gaming site used by many independent game producers to distribute their titles. To make the executable compatible for Steam, the team used Greenworks.js, a library that integrates JavaScript applications with Steam. Without that step, according to Arch, tasks such as saving player advancement and achievements in the cloud–to share with all the player’s computers – wouldn’t have been possible.

Insights from an Experienced Game Developer

Although he is modest about calling himself an expert, Arch has some valid advice for beginning game developers – starting with some things he would do differently. “I would better check the maturity of the technology I use,” he said. “Making games with HTML5 is really, really odd, but it has some good points. Debugging is easy, and we do not have to compile a new game each time we change something. That is very cool to quickly see what you're developing.”

The drawback is that the HTML5 infrastructure is not fully complete. The team often grew frustrated trying to make a high-end game with 3D graphics, trying to track several events happening simultaneously on the screen. “The HTML5 technology was not mature enough to easily do what we needed,” Arch said. “Getting it all working smoothly was very challenging.”

Arch also stressed being sure your team has the right skills to get all the way through the project. “What we missed,” Arch said, “were the marketing experts. We didn’t have someone that knows how to present a game to people. Fox and I were really into the *making* of the game. Once it was time to talk about it, we were not the people to do that. That's something people should know before starting a new project: know how to *sell* it when the time comes.”

Still, there has to be something to sell first, and that requires a team with solid technical skills. The advice from the Morphiks team is to stick with what you know, and don't assume you can pick up everything as you go. No matter what, Arch stressed, you’ll be stretched to take on new abilities. “It's not even a question; it's the normal process,” he affirmed. That may mean opening up the documentation and going through it, watching videos, or downloading demos and sample code. Arch’s team did all those things. “We actually enjoy reading manuals,” he added.

Stick to Your Instincts

Arch credits winning their award for Best Action Game to staying true to their vision, regardless of all the advice they received. Players liked the very simple interaction of *Synchrom*, but it sometimes felt like new players were suggesting strange fixes or additions that would dilute the game’s appeal. When showing something new to gamers, Arch felt it was important to listen, but to stay strong. “You don't know the background of those playing your game and giving you suggestions,” he pointed out. “It's important to read between the lines and understand what your early players and testers are really saying. Do what you like to do. Don't just create things that others want you to make.”

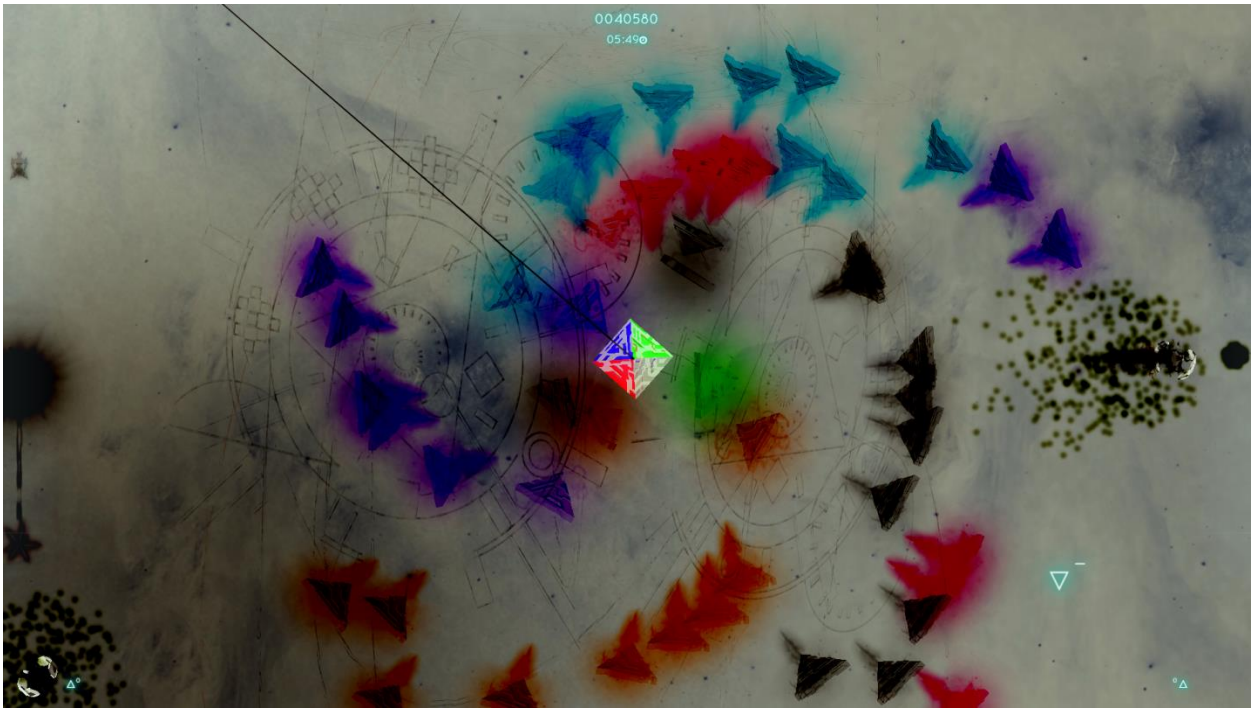

Figure 4. Morphiks switched between renderers to apply the in-game effects in the infinity stage. Switching gave them a plain, simple renderer for the rest of the graphics.

At the end, it's the results that matter. "We are really happy with what came out," he said. "This game is really us, it's what we are, and what we want to share with people. Even if it's not a perfect product, in terms of a commercial title, it's more ***us***. We think we're on the right path. Winning an award in the 2015 Intel Level Up contest is a nice validation for us."

The Morphiks team finished a major update in October 2015, based on feedback from players and known technical issues. Continual updates to that project are planned, and an Android* version is on the horizon. At the same time, the team is definitely starting a new project. They don't want to say too much about it right now, because it's still in the early development phase. It will be a multi-player game and probably use the same engine they developed for *Synchrom*, but their new title will allow for more players, offer online play, and work well on more mobile devices.

That's a smart approach, because mobile gaming is smoking hot right now, across laptops, 2 in 1 devices, smart phones, tablets, and other portable products. According to a recent report released by the research firm Newzoo, mobile video games could climb to an expected $30 billion in annual sales by the end of 2015, with growth rates in the major markets reaching 50 percent or more. Yet gaming on smartphones has stalled; what's driving that expansion is gaming on tablets, and a unique new title such as *Synchrom* can't help but stand out in that field. That gives Morphiks plenty of motivation to push forward.

"We learned a lot getting to this point, and we're excited to keep going," Arch said. "There are several Intel tools out there that we look forward to learning; we've never been afraid to try something new.

One in particular that really gets our interest is the Intel XDK, which offers an easier path to HTML5 app development. That one's at the top of our list."

To get the latest developer graphics guides, code samples, tools, and support, visit the Intel® Game Developer Program

Resources

Check out *Synchrom* on Steam: http://steamcommunity.com/app/393740

Learn more about Morphiks: http://www.morphiks.com/

Get the details on the 2015 Intel® Level Up Contest:
https://software.intel.com/sites/campaigns/levelup2015/