Alexander Kuvshinov used to dream of creating something more than a simple Adobe Flash* banner. But a string of low-level technical jobs at an architectural firm, a printer, and an advertising agency kept his creativity hidden. Then he teamed up with Andrey Sharapov and Roman Povolotsky, two veterans of the Russian game industry, to found AuraLab. Their first title, *Karma.Incarnation* 1, has already won major awards for art, design, sound, and storytelling; players around the world are awaiting this Russian team’s advance from a demo to a commercial game.

*Karma.Incarnation* 1 is a 2D adventure game in a whimsical, psychedelic world reminiscent of the 1968 cartoon classic *Yellow Submarine*, or possibly inspired by 60s artist Peter Max. In the game, the hero, a blob-like tentacled character named Pip, must learn from his environment and understand the cause-and-effect logic behind what happens as he travels forth, solving puzzles and conquering mazes. When Pip takes ineffective actions, his appearance is altered in a threatening way; when he takes desired actions his appearance becomes more appealing.

Getting game development to its current point has been a challenge for the AuraLab team; just moving from doodles on paper to digitized animation took several months. Advancing to a playable demo was another challenging push, and even more hard work is on the team’s horizon to bring this game fully to market. But with major wins at the 2015 Intel® Level Up Game Developer Contest, it’s worth pulling the team aside for a deeper dive into some of their secrets.
What Inspired This Team?

The inspiration for Pip, the main character of *Karma*, actually came from an art contest Kuvshinov had entered for creating a design for the back panel of a tablet. The contest initially intrigued him because the winner was to receive an Apple iPad* with the winning design printed on the tablet’s back. Kuvshinov started played around with a black Apple logo, using the bite to form a kind of sideways mouth.

He quickly had a whole herd of these creatures hanging around on his computer—he’d sketch some random blot, throw in some eyes and tentacles, and play around with that. “I didn’t win the main prize, but by then it didn’t matter much, because I had my new creations to occupy my mind,” he said.
His cartoon grew in sophistication, but in 2013, the agency where he worked disbanded, which turned out to be a blessing in disguise. He and his friends soon formed their own gaming company – AuraLab. They set right to work bringing Pip to life.

**Leveling Up Leads to Pre-Production Rush**

After the team finished their pre-alpha engine based on Adobe Flash they had three levels, or “locations” in the game. Pip could move between those locations, and the game had some simple interactions as well. “Still, we had a long way to go,” Kuvshinov remembered.

Fortunately, the team already had decent design and iterating skills based on their deep experience in technical positions. When asked what skills were most important for the team at this point in time, Kushinov explained without hesitation that their prior familiarity with the entire game-development process was key. “We knew what skills we needed for our team, and we had them,” he said. Those abilities included technical skills, experience with development processes, the ability to iterate and test, and a strong background in design.

The team worked from home, using only free software. Google* Docs* was their main collaborative tool during the pre-production phase of the project. For task tracking as they plowed through more dependent work, they used Redmine, which they installed on a virtual server that they purchased. “We didn’t buy any hardware—we all have PC’s, notebooks, and iPads, and that was enough,” Sharapov said. “And after we built our demo on Unity*, we

*Google, Docs, Unity: Supplied
actually won a Unity Pro license as the Best Unity Game at the White Nights ’15 conference, so we never had to purchase that license, either.”

Kuvshinov saw an announcement for the Intel® Level-Up Game Developer Contest on http://www.promoterapp.com. Meeting the qualifications and with a decent prototype demo that would take advantage of the touch screen and adaptability of 2 in 1 devices, he entered the contest. “Then, I forgot about it,” he admitted. “I didn’t expect anything, to be honest. We were all were very surprised to hear about the prizes we won.”

So unlike how some teams have to put in a mad scramble to produce a demo worthy of submission, AuraLab really got busy after its win. The team plugged into the global indie game-developer circuit and rolled up their sleeves. “The main work began after we received the message that we won,” Sharapov said. “That was when we started to prepare the build for the exhibition at PAX Prime, as well as to prepare the Steam* version.”

![Figure 4: AuraLab uses intense shading and surreal art to create a unique mood for players.](image)

**Figure 4: AuraLab uses intense shading and surreal art to create a unique mood for players.**

**Unity*—The Engine of Choice**

One early decision the team debated was which game-development engine to choose. Two popular engines are Unreal Engine* 4 and Unity Engine, and while they both were strong contenders, AuraLab chose Unity.

“Unity allows us to make cross-platform games easily,” Sharapov said. “We already had some experience with Unity before, and that helped. Also, it is much easier to find good Unity programmers for the project in Russia.”
Once AuraLab converted to the Unity Engine, they initiated some serious game design sessions and developed the real demo that the public can play. They used the 2D toolkit plugin found in Unity 5, but they had a few challenges to overcome. So lead programmer Yuriy Kuptshevich created several tools to speed up the following tasks:

- Upload and unload atlases in runtime.
- Upload a series of atlases sequentially.
- Calculate the number of textures that can fit in an atlas.

Kuptshevich also conducted several experiments with graphics compression, but he quickly learned it was problematic to try uploading a texture from a *.png file asynchronously. The Unity team has promised him a fix in version 5.3.

**Helpful Hints—Lessons Learned**
Because the team so recently struggled to get their own game noticed by the broader public, they know the challenges that developers face. In that spirit, and maybe to build up some good karma of their own, Kuptshevich is eager to share the following tips and tricks with his fellow indie developers:

How to bind the zoom of particles and objects in a scene:

BorderLeft and BorderRight – pivots the edges of the object

```csharp
public void ScaleParticles ()
{
    float size = Vector3.Magnitude(Camera.main.WorldToViewportPoint(BorderLeft.position) - Camera.main.WorldToViewportPoint(BorderRight.position));
    _particleRend.maxParticleSize = size;
}
```

How to create a deferred-in-time lambda expression via the delegate of Action or call them in the loop for a set time, or both:

```csharp
public void InvokeWithDelay (Action method, float delay)
{
    StartCoroutine(InvokeMethodWithDelay(method, delay));
}

private IEnumerator InvokeMethodWithDelay (Action method, float delay)
{
    yield return new WaitForSeconds(delay);
    method.Invoke();
}
```
Kuptshevich’s best advice is this: “Keep it simple. The main thing is to not do complicated things if at all possible.” He also recommends that developers look for places where a tool can help with repeating processes. His team expanded the Unity editor with their own tools, doing everything possible in an automated way. “You might think you will spend too much time on the tool at first, but automation will make your life much easier and speed things up quite a bit when you are deep into the main part of the development process.”

The team would do a few things differently given the chance, Kuptshevich admitted. Still, experience is a great teacher. “This was not the first game for any of us, “he said. “We didn’t make any critical errors in the development process because we all have a lot of experience in the pre-production phase. That’s where the planning and the design decisions are very important.”

AuraLab made a great, straightforward game with fewer locations but more intensive gameplay. That resulted in a shorter and cheaper production plan. “The moral is, if you can split your work into sections and do that first part faster, with the same or better quality, just do it,” Sharapov said. And once you make a decision, he says to stick with it—“with no mercy.”

**Live Demos Provide Positive Feedback**

Without a sizable testing team behind them, AuraLab relied on in-person demos at gaming events, conferences, conventions, and more, for testing their game at various stages. “Our testing methodology was to run each scene separately,” Kuptshevich said. “In a game with a lot of related scenes, it is also important to be able to easily test from any location in the storyline. It sounds trite, but it is necessary to remember about it from the very beginning, and development in general will be easier.”

To describe the cutscene script the team decided to use coroutines inside other coroutines. These components generalize subroutines for nonpreemptive multitasking, by allowing multiple entry points for suspending and resuming execution at certain locations. The choice gave them the flexibility to manage asynchronous events, after writing some custom code to make it easier to manage events associated with triggers in animation.

They used Mecanim’s Animation State Machines for logic. According to the Unity documentation, “Mecanim’s Animation State Machines provide a way to overview all of the animation clips related to a particular character and allow various events in the game (for example, user input) to trigger different animations.”(For more details, the team refers the reader to Code Project, which has a link to Simon Jackson’s book on Mastering Unity 2D Game Development, which covers state machines in depth. You’ll also find a link to his sample project code with state machines.)
The AuraLab team stressed that developers should take the time to create custom inspectors for commonly used classes, as they add a lot of power and flexibility to most workflows. “In the long term, it will save you time,” Kuptshevich said.

As for feedback directly from players, Sharapov said that users quickly identified some annoying bottlenecks in gameplay that had to be fixed. As a result, they reworked parts of the demo, making it easier for the user to pass through from beginning to end. However, more difficult riddles are already in development and are sure to please the conference crowds.

![Figure 5](image)

*Figure 5: The AuraLab team succeeded in creating scenes in Karma that are whimsical, engaging, and beautiful.*

*Karma* has an interesting, original soundtrack, with engaging effects and a soothing musical flow. The team credits that to their collaboration with the Zmeiraduga band, whose beautiful, improvised music has developed a bit of a following. The band’s 100-percent improvised music means a completely different concert experience each time out. To create the rest of the sound effects, Kuvshinov used a microphone, distributed free samples, and a vivid imagination.

**Multi-Platform Success**

Because the Unity Engine offers flexibility with its output targets, AuraLab had no trouble creating demos for multiple platforms. They currently support Microsoft Windows* 7/8, Windows Vista*, and Windows XP* SP3; Mac*, and Android*. The team is planning to develop for AppleTV*, Microsoft Xbox*, and Sony PlayStation* after they release the first commercial version on PC, Mac, iOS*, and Android.
“The Unity Engine allows us to create these multiple releases without too much effort. However, we have to refactor our game several times before we have a stable build on multiple platforms,” Sharapov said. All the different versions have unique features, so it helps to use a game-development engine that knows those subtleties.

“After several experiments and months of refactoring the code, we are to the point where we have a stable architecture for the project and a solid development process that allows us to make builds for each of our supported platforms at one time with a minimum of hand work,” Sharapov said.

One bit of planning really helped at this stage—the team always planned for their gameplay to work on both mobile and desktop versions. They knew they would have to accommodate many different types of user controls, from gesture to touch to point-and-click. “We planned the gameplay from the very start so we could handle both mobile and desktop versions,” Sharapov said. By planning for multiple versions up front, they avoided coding in bottlenecks that would take longer to resolve down the road.

For example, the team kept in mind the idea that users might not have a mouse or may only use a touch screen. One place that showed up was when they planned out a “hint” icon. They saw in testing that they couldn’t show hints by requiring the user to mouse over an object. They realized also that they couldn’t draw small clickable elements too close to each other, because they needed big tap zones for mobile phone users.

Testing sessions have frequently led to fixes they would never have considered. “We learned how the camera position and zoom at the cutscenes for a mobile phone is different from those for the desktop version,” Sharapov said. “We are learning fast.”

The team inserted a hint icon on the bottom-right corner, building it so that when a new hint is coming in, the lamp on the icon begins to shine. Unfortunately, when they tested the hint on an iPad, they learned that most of the users didn’t see the hint icon because of the natural tendency to cover that area with a hand when holding the tablet. Their response was to add additional support for the hint icon right into the game viewport. Now the lamp shines over the head of the main character AND in the bottom-right corner.

**Next Steps**

The future looks bright for Pip and AuraLab. At the time of this writing, no announcement date has been set yet for the full download of *Karma* across all four main platforms, but the team is hard at work. Already they are planning a second episode of the game to fully explore Pip’s incarnation. They’ll continue the storyline and come up with more riddles, puzzles, and mazes, and it should be a fun ride. And based on their willingness to share tips, tricks, and codes, maybe they’ve built a little positive karma for themselves.
Resources

YouTube* trailer: https://www.youtube.com/watch?v=t=1&v=d_PeeemZIBA

Kobe-channel with six unique mini-videos: http://coub.com/karma.incarnation1

Karma demos: http://karma-adventure.com/demo.html

Unity Engine download: https://unity3d.com/get-unity


Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2015. Intel Corporation. All rights reserved.