Shadertoy

Learn to Create Everything In a Fragment Shader

Pol Jeremias & Inigo Quilez

SIGGRAPH Asia 2014, December 03 – 06, 2014, Shenzhen, China.

ACM 978-1-4503-319501/14/12
http://dx.doi.org/10.1145/2659467.2659474

1. Introduction

1.1 Course Overview

This course will introduce innovative, artistic and creative ways to use fragment shaders in your web browser using Shadertoy. The attendees will learn about techniques such as raymarching, procedural content creation (texturing, modelling and animation), image compression and volumetric rendering, by using live coding. The algorithms covered in this course are state-of-the-art and they are currently used in different industries, such as interactive entertainment or movies.

The following notes are intended to be a high level description of the topics covered in-depth during the course at Siggraph Asia 2014.

1.2 Prerequisites

The intended audience is those professionals or students with interest in Computer Graphics that want to learn the intersection between code and art. This includes game developers, filmmakers and artists.

The course requires an intermediate knowledge of: basic computer graphics, shading/lighting models, shading languages (GLSL, HLSL or RSL), algebra and geometry.

1.2 The speakers

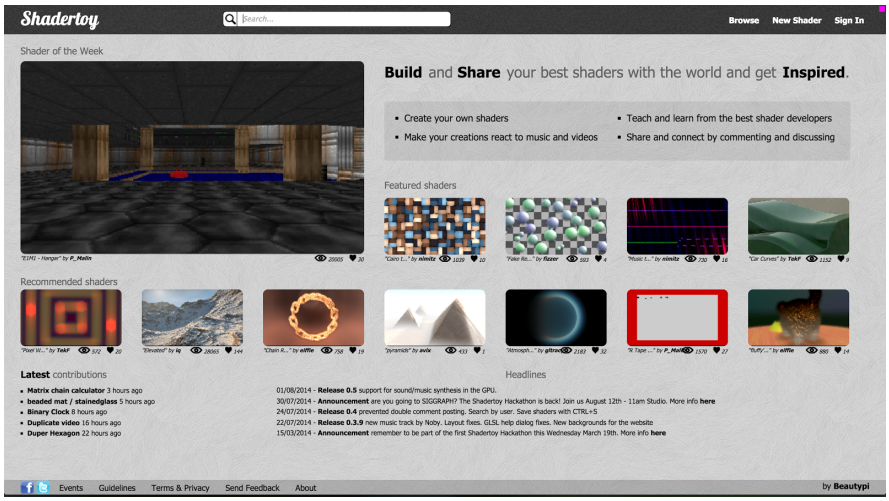
Inigo Quilez is fascinated with the potential of using code and maths to build visual beauty. After many years in the demoscene, Inigo worked in virtual reality and real-time rendering. Today he's employed at Pixar Animation Studios, inventing techniques, drawing and creating procedural imagery.

Pol Jeremias is passionate about technology and art. He uncovers the intersection with engineering and his imaginative mindset. He holds a Master's in CS from University of Southern California. He worked on Star Wars 1313 at LucasArts, and today, he's part of SoMa Play Inc, a AAA game studio.

Both founded Beautypi, which aims to bring high quality interactive and reactive graphics to the world of entertainment, culture and business. Shadertoy is their first toy.

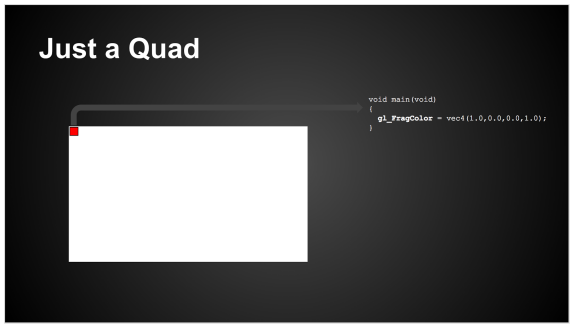
1.3 Introduction to the Platform

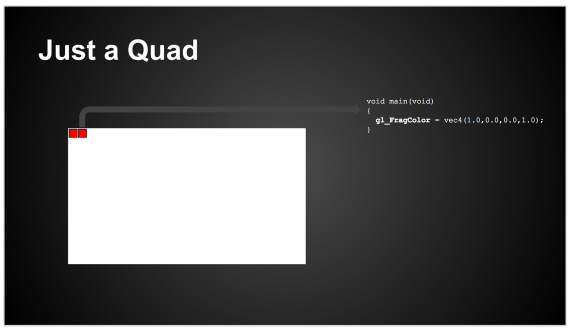
Shadertoy is a web tool that allows developers all over the globe push pixels from code to screen using WebGL. In less that two years, our users have created thousands of fragments shaders that explore a wide variety of real time graphics algorithms. Shadertoy is also a social platform - a place for professionals and students alike to learn and teach about visuals, interactions, reactivity, procedural modeling, GPU internals and shading.



The challenge of Shadertoy

Shadertoy has one challenge, the contributors can only write a fragment shader that is applied automatically to a quad. This restriction makes the creation process very minimalistic but, at the same time, it has sparked new ways to create unimaginable content.



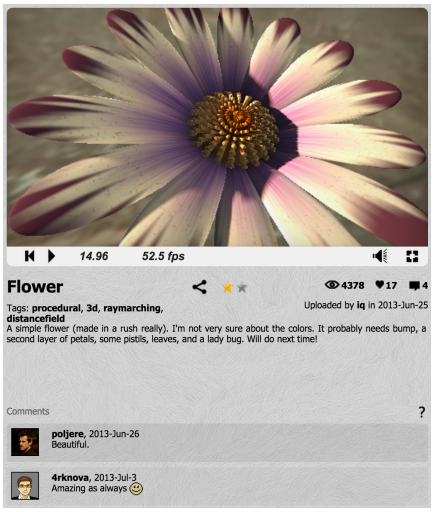


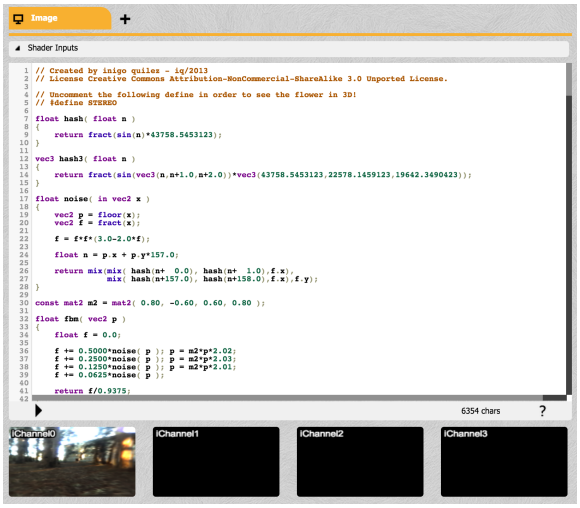
Code Editor and Visualizer

In Shadertoy the creation environment is split between the visualizer (left picture below) and the code editor (right picture below).

The visualizer has the usual options such as play or rewind, but it also contains the description and other relevant meta data of the shader.

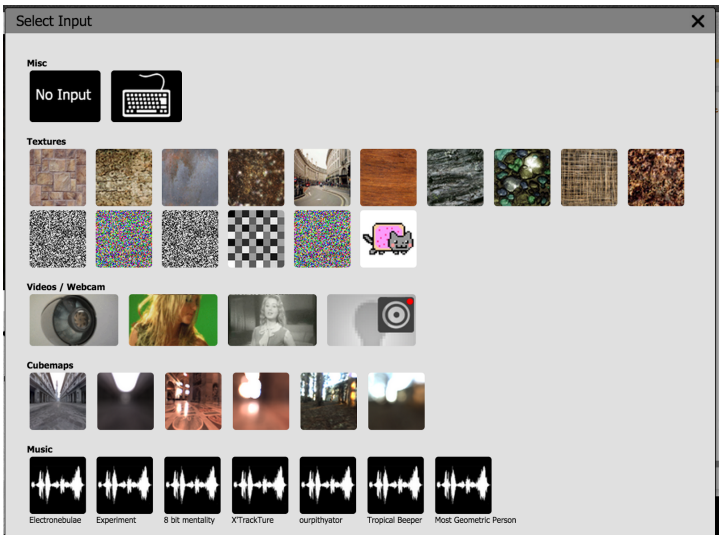
The code editor has syntax highlighting and it also shows compile errors and other relevant information. The user can also see the “Shader Inputs” available in the shader, or create a new “Pass” to add audio to the shader by pressing the “Add” button.





Resources

Every shader in Shadertoy can use (at will) a limited amount of textures, sounds or videos. Once the user has selected one in the boxes below the editor then they are accessible via texture fetches.



Community

Shadertoy is a community with thousands of users. In order to publish content users need an account, which is free. Users can check other users' profiles and navigate through each other's creations. On top of that, every shader includes a comment section where content creators can answer questions from the community or receive feedback.

2. Thinking implicit, 2D: non polygonal shapes

2.1 Introduction to implicit thinking

Contrary to the natural way of constructing images by explicitly placing geometry or color in different positions over space and time, Shadertoy encourages (if not imposes) the use of implicit methods to achieve the same. Even content typically built with procedural methods tend to be explicit, it is simpler to understand and translate into a series of routines or procedures (hence *procedural*) that a computer program can follow. Examples of these are L-Systems, subdivision based geometry, cloth simulation, terrain erosion, and also, in general, all the regular modeling and texturing/painting pipelines used in production of games or movies. Only a few popular algorithms lead to an implicit description in a natural manner, such as fractal rendering or raytracing.

Hence, when in need of a complex animation model in Shadertoy (say a swimming dolphin, or an obstacle avoiding spaceship) users need to define its shape and modeling, texturing and animation in an implicit manner, despite the lack of culture and tradition around implicit approaches.

So, in order to create rich content in Shadertoy, more often than not users need to wrap their heads around the new way-of-doing, which requires imagination and creativity. Sometimes, they might even find themselves in places where very few or even no researcher has even been before! Other times, it might help them approach a simple challenge from a very different perspective, giving them a deeper understanding of the problem.

This section will quickly review some of the basic principles of implicit shaping (forms for modeling, colors for texturing/surfacing and movement for animation).

2.2 Coordinates and color gradients

This first shader will show how to create a colo gradient to ensure the rectangular concepts o coordinate systems, value normalization and color representation are understood.

This is a mere introduction, and no time will be spent in anything but very short explanation and coding.

https://www.shadertoy.com/view/4ssXWl

2.3 Shaping

How to go from rectangular/cartesian coordinate system to a circular coordinates? In this example the audience will see how to define circles and radial patterns.

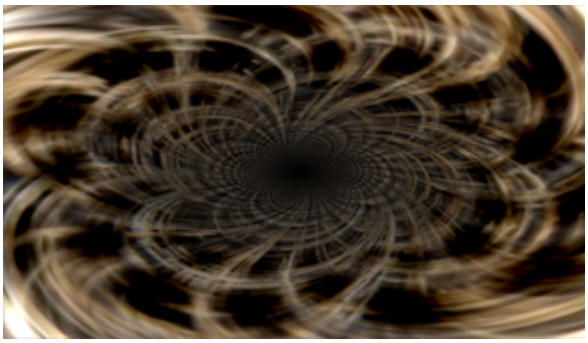
Smoothstep will be used to delimit plane regions and colorize them differently. This will serve also as a very quick review of the concepts of amplitude frequency and phase in sinusoidal signals.

https://www.shadertoy.com/view/4dXGzn

2.4 Texture mapping

While the use of texture mapping defeats the purpose of building everything procedurally Shadertoy is not about procedural content creation per-se, but about implicit thinking.

Using textures can be an easy way to add detail in a simple way with high quality filtering (as opposed to pure procedural patterns).



https://www.shadertoy.com/view/Xdf3Rn

2.5 Animation

Animation in Shadertoy needs to be implicit too.

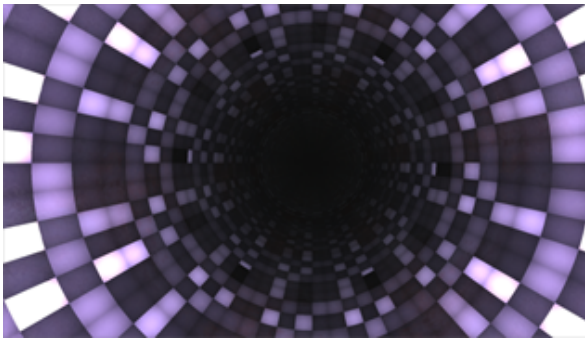
The lack of access to the frame/time history makes things such as physics integration or simple movement logic impossible. Hence, everything that is animated in Shadertoy must be a function of time and nothing else but time. No IIR filters can be implemented, no forces can be applied to a mass to bend it over an obstacle. Complex movement that mimic obstacle avoidance for example must be achieved only as a function of time.

https://www.shadertoy.com/view/XsfGRn

2.6 Reactivity

In order to make Shadertoy more interactive and fun, we do allow pre-made sound waveforms be used to feed the implicit shaders.

A few soundtracks are provided in the site from which the user can pick. At render time the actual waveform and frequency spectrum of the soundtrack are extracted and copied to the GPU thru the use of textures for the shader to consume.



https://www.shadertoy.com/view/lsBSRR

2.8 Adding other media to your creations

Besides textures and sound, Shadertoy allows the use of video, mouse and keyboard input and time/date data.

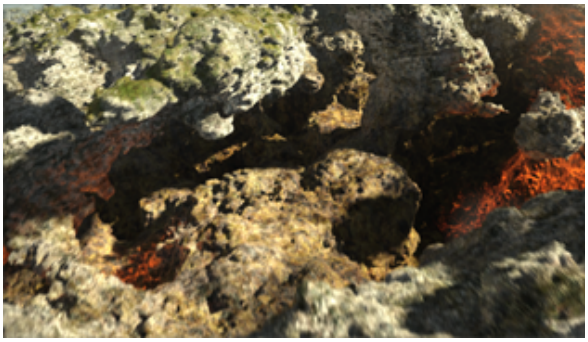
https://www.shadertoy.com/view/4dl3D7

3. Thinking implicit, 3D

3.1 Raymarching

Raymarching is the most popular technique to get 3D scenes rendered in Shadertoy. It shares the simplicity and implicit nature of raytracing, but it provides a much more powerful way to describe 3D shapes and lighting.

This course will spend a good amount of time in this area to make sure the attendees understand the technique, as well as, the different parameters that control the quality and speed of the images, such as number of steps, step size multipliers, distance-field based marching vs linear, etc.



https://www.shadertoy.com/view/XsX3RB

This section will also cover how the implicit definition of the surfaces leads to the idea of a field and how a normal can be extracted for it for lighting.

The following sections will be the foundation for the rest of the course, so time will be invested in giving a solid understanding of the raymarching based imagery creation process.

3.2 Modeling

Modeling of implicit distance fields can be converted into an explicit problem by means of the use of primitives and operators.

Primitives define simple shapes in an analytical closed form. Spheres, rounded boxes, helixes, etc are good examples.

Domain operators allow to scale, bend, twist and repeat shapes. Range operators allow to combine and displace and deform shapes.

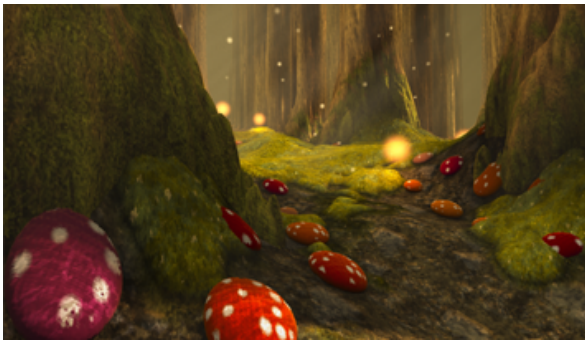
Many of these operators don't need to be mathematically perfect as long as they fulfill some conditions, making the process of modeling very exploratory and intuitive.

https://www.shadertoy.com/view/ldl3zN

3.3 Displacement

Displacement is probably the most simple of the operators and it's easy to understand. It's power comes with tradeoffs (such as breaking the conditions of distance metric that need to be compensated with smaller step sizes) that need to be understood.

However its a very powerful technique. For exampl vertical cylinders can be displaced with an exponential at their base and some global noise i order to craete tree trunks.

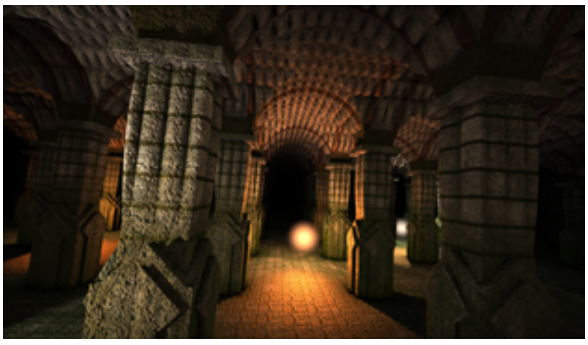


https://www.shadertoy.com/view/XsfGD4

3.4 Domain Repetition

Building huge 3D sets (or infinite) can sometimes be challenging depending on the technology used In the case of raymarching, domain repetition is one way to achieve this with one single line of code, in a very cheap way.

There are rules that need to be followed for th technique to work, but once these are understood, it's a very powerful tool to have in the toolbox o implicit procedural modeling.



https://www.shadertoy.com/view/lsf3zr

3.5 Shape Blending

Shape blending, similar to displacement, is a range operator. However this one can be used t composite shapes in a way similar to a union operation which produces C1 continuous surfaces while it behaves quite as a union in the regions o space which are close to only one of the surfaces, it behaves in as a smooth inflate operator in the areas what are in the proximity of both source shapes.

This technique is very useful to create shapes that are smooth and organic.

https://www.shadertoy.com/view/MsXGWr

4. Thinking implicit: stateless animations

4.1 Procedural animation

Game and film production often rely on incrementa animation systems, where the final position of the elements is determined by the integration of instant forces and velocities over time. Such systems can easily react to the relative position of the differen elements in the scene and create rich behaviors Player and enemy positioning in a game ot physics simulations are such examples.

In Shadertoy the time axis of the procedura animations is implicit as well, just like the image. No data is passed across frame boundaries, making it impossible to implement integrators or incremental object positioning.

https://www.shadertoy.com/view/4dSGW1

Instead, all shaders need to find implicit formulas or algorithms that produce plausible animations, which can be challenging. However, obstacle avoidance, behaviour simulation and physics can be achieved by different means. The following sections will explain how to address some of these challenges.

4.2 Blending animation curves

In the same way the implicit modeling of surfaces can make use of a hierarchical approach where simple shapes are combined and blended together to construct higher complexity surfaces, implicit animations can be enriched in a similar way.

Given to implicit animations, for example a underwater swimming and a out-of-water jumping animation, we'll see one example of blending both in a smooth way such that different behaviours car be combined into a new single animation function.

https://www.shadertoy.com/view/4sS3zG

4.3 Inverse Kinematics

Inverse kinematics (IK) is a well known olc technique used for letting the computer figure ou some of the animation posing of a system, giver some constraints. This implies making computations that usually happen as a step before rendering starts.

Since in Shadertoy the concept of pre-render doesn't exist but everything necessary to make the image happens at once, one needs to perform the full character IK computations once per pixel (if not more).

https://www.shadertoy.com/view/Mss3zM

4.4 Simulating Physics

Physics integration is the canonical example of an explicit procedural process, which of course is not possible to implement in Shadertoy. Hence, physics have to be solved in a different manner.

By simplifying the rules of the system (no collisions between the bodies in it) we can find analytical solutions to the full motion of the bodies in the system and create physics based animations.

https://www.shadertoy.com/view/MdB3Rc

5. Pro-Tricks

5.1 Depth of Field

In the context of raytracing and raymarching, depth of field (DoF) or lens defocus can be trivially solved by supersampling the rendered image and taking the sampling of the lens (ray origin and direction) However the technique becomes expensive very quickly for realtime rendering.

Luckily, in the last months the users of Shadertoy have figure out ways to take advantage of the distance field representation of most of the implicit modeling used in Shadertoy.

https://www.shadertoy.com/view/Md23z1

5.2 Anti-Aliasing:

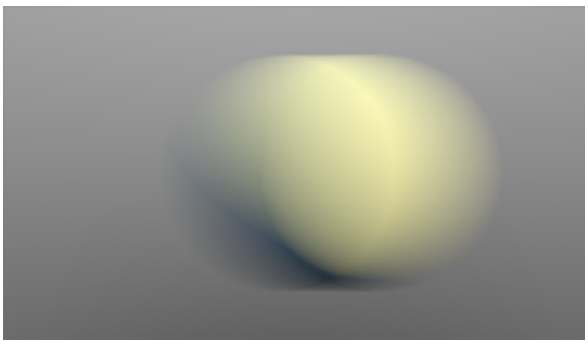
As an extension of the technique used for distance-field based DoF, antialiasing can be solved as well by exploiting the properties of the implicit fields.

https://www.shadertoy.com/view/4dS3Rh

5.3 Motion Blur

Motion Blur is also traditionally solved by multisampling. However, it's sometimes worth rethinking a problem and try to figure out if there are faster ways to solve it.

In some particular cases Motion Blur can actually be solved analytically, which is surprising, but also comes with its own set of problems and challenges.



https://www.shadertoy.com/view/MdB3Dw

5.4 Soft Shadows

Soft shadowing (with penumbras) is anothe example of a solved problem in computer graphics for ray based rendering algorithms by means o integration thru multisampling.

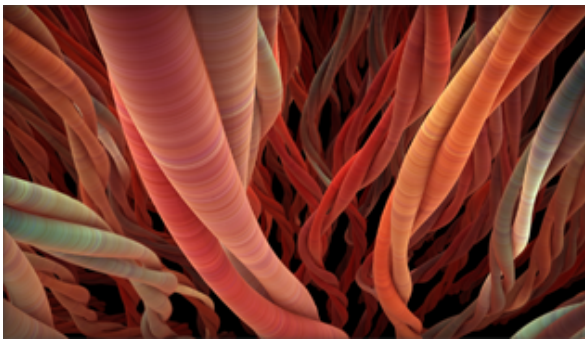
However, once again distance fields provide enoug global information about the scene that creating convincing soft shadows can be achieved by castir one single shadow ray.

https://www.shadertoy.com/view/ldl3zl

5.5 Ambient Occlusion

Ambient Occlusion is a cheap way to bring global illumination look to renders without solving the ful global illumination problem.

This section will show how distance fields and implicit procedural can be used to create 4 differen ambient occlusion algorithms that are fast enough to be rendered in realtime.



https://www.shadertoy.com/view/XsjXR1

5.6 Atmospherics

Atmospheric effects play an important role in conveying the sense of scale in a scene. Although the tricks in this section apply equally to a regular rasterization based renderer, Shadertoy's live coding feature makes it a nice framework to explair some of the artistic possibilities and tricks to create interesting fog and scaterring cues.

https://www.shadertoy.com/view/MdBGzG

6. Q&A