Interactive Evolution of Camouflage

Technical Report IEC-2021-1

August 21, 2021 version 1

Craig Reynolds
cwr@red3d.com
https://www.red3d.com/cwr/

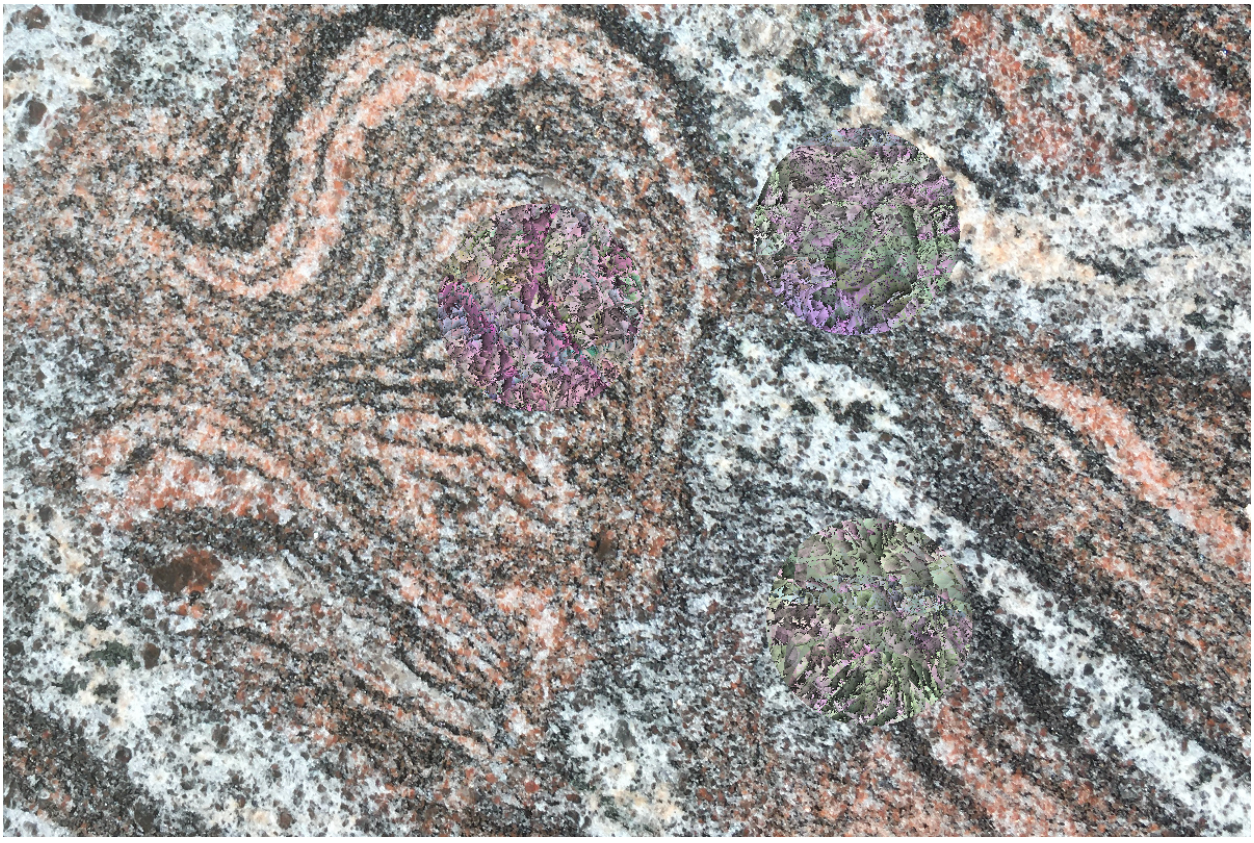

Fig. 1: photograph of a polished granite counter-top in bright sun, overlaid with three synthetic “prey,” circular samples of evolved camouflage texture.

Abstract

A simple computer simulation of the evolution of camouflage in nature is described. Several examples are given. The model is based on a game like concept with a human "player" in the loop to act as a predator. This is a reimplementation of a technique first implemented in 2009??? [???]. The basic concepts are the same between the earlier work and this new implementation. The point of this work was to write a clean modern software infrastructure, to verify the old results could be reproduced, and to then move on new topics in this general area of research.

Background

Description of software

These experiments use three primary software components. First is a library for procedural texture synthesis called *TexSyn*. Second is a general purpose engine for optimization via evolutionary computation, called *LazyPredator*. And third is a simple interactive application that uses those components to make a game-like simulation where camouflage evolves in response to selection pressure from a predator played by the human.

App

That app is defined by a c++ class called *EvoCamoGame* and is invoked from the unix style command line (maxOS *Terminal* app) with an executable called `evo_camo_game`.

Procedural texture synthesis

TexSyn is a c++ library for procedural texture synthesis. It defines objects (instances of a class) called *Texture* which represent an abstract concept of color textures. Common digital images are defined by a rectangular array of pixel values each representing a color. *TexSyn*'s textures are purely procedural. They are defined by code. A Texture is an infinite 2d plane. Its basic operation is to return a color for any location on that plane, specified by x, y Cartesian coordinates as floating point numbers. In c++ terminology, *Texture* defines a "virtual member function" called *getColor()* which maps a *Vec2* position into a *Color* value.

Texture is a c++ base class on which *TexSyn* defines about fifty specialized types that "override" the *getColor()* function. A couple of these define a texture from numeric parameters, such as *Uniform* which is everywhere the same color. Most of *TexSyn*'s classes are *texture operators* which take one or more *Textures* as parameters. Complex TexSyn textures are nested expressions — "trees" — that describe combinations of many texture operators. Typically these trees have between 100 and 200 operators (plus various numeric values at the leaves) but there is no explicit upper bound on size.

??? example ???

??? links to blog and GitHub repository ??

optimization

…

evolutionary computation

…

genetic programming

…

OpenCV

…

old:

This is an alpha test version of software for “interactive evolution of camouflage” as described in my 2011 paper: Interactive Evolution of Camouflage. This 2021 app evo_camo_game is built from new components: the TexSyn library for procedural texture synthesis, and the LazyPredator library for evolutionary optimization via genetic programming. For more information see the development blog for TexSyn.

Very quick overview

This program is a crude simulation of the evolution of camouflage in nature. There is a predator-prey system. Software for texture optimization plays the part of an evolving population of camouflaged “prey.” The human user serves as a predator hunting its prey with vision. This can be seen as a sort of minimalist “game” or a human based computation. The app displays a window with a random portion taken from a given set of photographs. Over that background are drawn three randomly positions disks of synthetic texture—three “prey.” The human user/player/predator then decides which of the three textures/prey is most conspicuous or least well camouflaged, indicating their selection by clicking/tapping on that prey. The window will go blank then display the next step. Runs typically consist of 1000 such steps or more. They can be stopped at any time. Results are currently saved during as image files, and texture “source code” in text files, as described below.