Universitat Politecnica De Catalunya, Barcelona-Tech.
Erasmus Mundus Masters of Science in Computational Mechanics

Report on Internship in
Institute of Earth Sciences Jaume Almera
(ICTJA-CSIC)
Spanish Scientific Research Council

By: Samadrita Karmakar

External Supervisor: Daniel Garcia-Castellanos
Internal Supervisor: Sergio Zlotnik, Josep Sarrate

21st June, 2018

PREFACE AND ACKNOWLEDGEMENT

I officially started my Internship in Instituto de Ciencias de la Tierra Jaume Almera (ICTJA-CSIC) on the 19th of January 2018. In English it means, Institute of Earth Sciences Jaume Almera. It is an earth science public research institute of the Consejo Superior de Investigaciones Científicas (CSIC). It was created in Barcelona, Spain in 1965 and it is considered among the top research institutes in Earth Sciences in Spain.

The basics of Internship objective was known to me much earlier in the month of September 2017. This was thanks to the efforts of Professor Sergio Zlotnik, the Master director. He introduced me to Professor Daniel Garcia-Castellanos of Structure and Dynamics of the Earth and Crystallography Department at ICTJA. Professor Daniel wanted change in the underlying code of the Augmented Reality Sandbox such that it would be more useful to geology students. The idea was to render the water flow simulation in the sandbox in a way that the regions that are vulnerable to erosion be rendered in a different colour. This would be the regions with higher flow volume and velocity.

This report expands on the working of the sandbox, the difficulties faced in executing the objectives, the improvements to the code and the results.

I thank Professor Daniel Garcia-Castellanos, Professor Sergio Zlotnik and Professor Josep Sarrate for all their help during the execution of this project.

CONTENTS

Table of Contents

1. The SandBox and the SaRndBox

The saRndBox project[1] has been developed in University of Davis California (UC Davis). It has been jointly developed by the UC Davis' W.M. Keck Center for Active Visualization in the Earth Sciences, together with the UC Davis Tahoe Environmental Research Center, Lawrence Hall of Science, and ECHO Lake Aquarium and Science Center. Its main developer is Professor Oliver Kreylos of UC Davis.

The project combines 3D visualization applications with a hands-on sandbox exhibit. It is an augmented reality (AR) sandbox. It allows users to create topography models by shaping real sand, which is then augmented in real time by an elevation colour map, topographic contour lines, and simulated water.

AR Sandbox uses a computer projector and a motion sensing input device (a Kinect 3D camera) mounted above a box of sand. The user interacts with the exhibit by shaping special "kinetic" sand in a basin. The Kinect detects the distance to the sand below, and a visualization an elevation model with contour lines and a colour map assigned by elevation is cast from an overhead projector onto the surface of the sand. As user move the sand, the Kinect perceives changes in the distance to the sand surface, and the projected colours and contour lines change accordingly. When an object (for example, a hand) is sensed at a particular height (~2 ft / 60 cm) above the surface of the sand, virtual rain appears as a blue, shimmering visualization on the surface below. The water appears to flow down the slopes to lower surfaces. The water flow simulation is based on real models of fluid dynamics, Saint-Venant shallow water equations, a depth-integrated version of the Navier Stokes equations.[2]

The sandbox was developed with the intention to raise public awareness and increase understanding of freshwater lake ecosystems and earth science processes. It has been created keeping in mind the requirement of it being self-contained. It should be able to be used as a hands-on exhibit in science exhibitions with little supervision.

1 Source: https://arsandbox.ucdavis.edu/

2 Source: "A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system" by A. Kurganov and G. Petrova

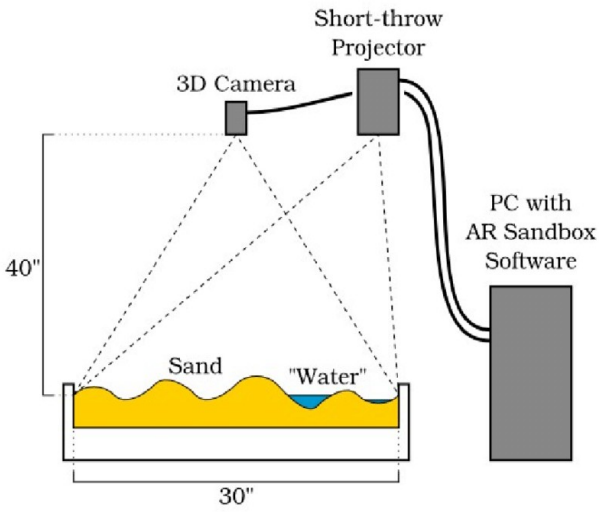

Fig 1a: A general setup of SandBox. Note that the “Water” like effect is from the Projector

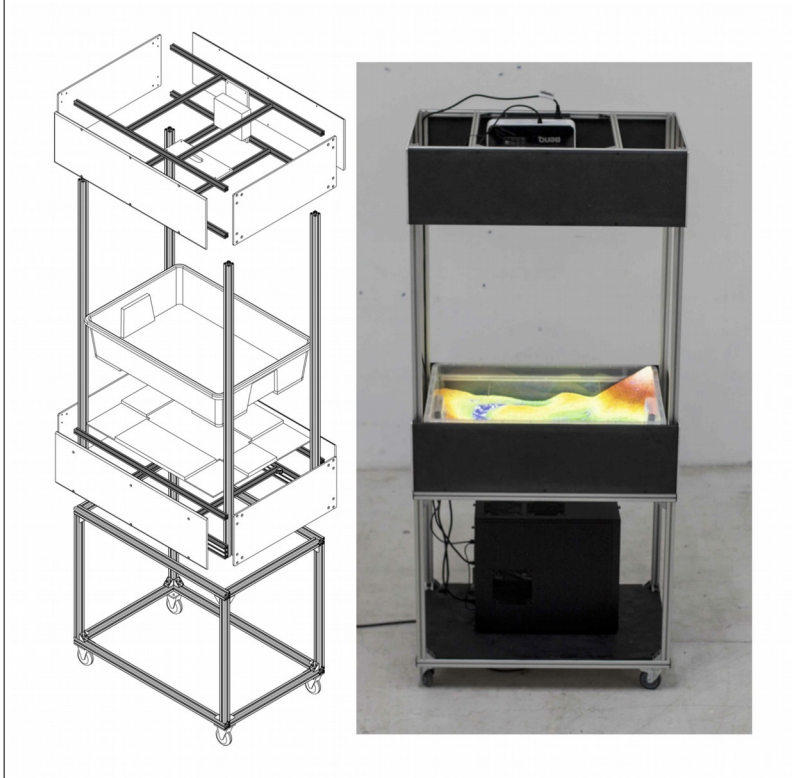

Fig 1b: Setup of SandBox in ICTJA-CSIC.
Designed by Raúl Nieves and Daniel Garcia-Castellanos

2. The Challenges of the Internship

The objective of the internship was to change the underlying code of the Sandbox such that areas more vulnerable to erosion can be identified during the running of the simulation. It was decided to represent areas with higher vulnerability to erosion with green colour and area with lower vulnerability with blue. The modification was motivated by the need to explain to students the process of erosion arising out of flowing rivers in a topography.
The Internship came with unique challenges which were the following:

- The SandBox is actively used in ICTJA for demonstration to students. Hence it was important to ensure that the setup remains functional after any changes are made.
- The software runs on GNU/Linux Mint which means that the anyone trying to change the code should have some experience in using such kind of Operating Systems.
- The software is heavily dependent on the output of the Kinect 3D camera to create the problem set and boundary condtion, that is solved by it and hence it generally cannot be tested without the actual SandBox Hardware.
- The software is written in C++. It is largely a visualizing software, so it heavily depends on Open Graphics Library API. Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. It is used extensively in the fields of computer-aided design, virtual reality, scientific visualization. OpenGL is managed by the non-profit technology consortium Khronos Group.[3] This meant that anyone changing the code has either to know about OpenGL API or learn it from Scratch.
- The software uses a combination of GLSL shaders to colour the surface by elevation and to add real-time topographic contour lines. At the same time, a water flow simulation based on the Saint-Venant set of shallow water equations, which are a depth-integrated version of the set of Navier-Stokes equations governing fluid flow, is run in the background using another set of GLSL shaders.[4] GLSL (OpenGL Shading Language), is a high-level shading language with a syntax based on the C programming language. It was created to give developers more direct control of the graphics pipeline without having to use some kind of assembly language or hardware-specific languages.[5] This is a part of OpenGL API.
- Due to the use of Kinect 3D camera to scan the sand surface and a Projector to project the results calculated by the software, the camera and projector needs calibration. Any newly compiled code may need both to be recalibrated.
- The implementation used in the software to simulate water flow follows the paper "a second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system" by A. Kurganov and G. Petrova. This had to be studied to understand the underlying code better.

3 Source: https://en.wikipedia.org/wiki/OpenGL
4 Source: http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/
5 Source: https://en.wikipedia.org/wiki/OpenGL_Shading_Language

3. The Implementation

The source code of SaRndbox is available in https://github.com/KeckCAVES/SARndbox. We have made changes in the version 1.6 of the software. In OpenGL the C like GLSL section of the program is sent to the GPU for compilation in real time during running of the main C++ code. In the C++ code it resides as strings. Hence changes can be made in GLSL code without any requirement of compilation of the C++ code. Any such change ensures that the SandBox remains functional, and any requirement of calibration of the projector or the calibration of the Kinect 3D camera is avoided. The changes can also be debugged and tested very fast. Hence all changes were limited to the GLSL section of the code.

The water colour is calculated, and applied to the rendered surface, in the "SurfaceAddWaterColor.fs" file residing in the "SARndbox-1.6/share/SARndbox-1.6/Shaders/" directory of the source code. The variable "quantitySampler" is a vector identified with red, green and blue components. The red component, *texture2DRect(quantitySampler,waterLevelTexCoord).r*, holds data on the elevation of the free surface of the water (bathymetry plus water column height over it). The green component, *texture2DRect(quantitySampler,waterLevelTexCoord).g* , holds data on the x momentum of the water (product of velocity times the vertical thickness of the water layer). The blue component, t*exture2DRect(quantitySampler,waterLevelTexCoord).b*, holds data on the y component of the momentum of water. The Erosion is directly dependent on this momentum.

To display render different colours, it was decided to interpolate the colour from the minimum momentum to the maximum momentum. The minimum momentum colour was to be rendered in blue and the maximum momentum in green. It may be visualized as the following.

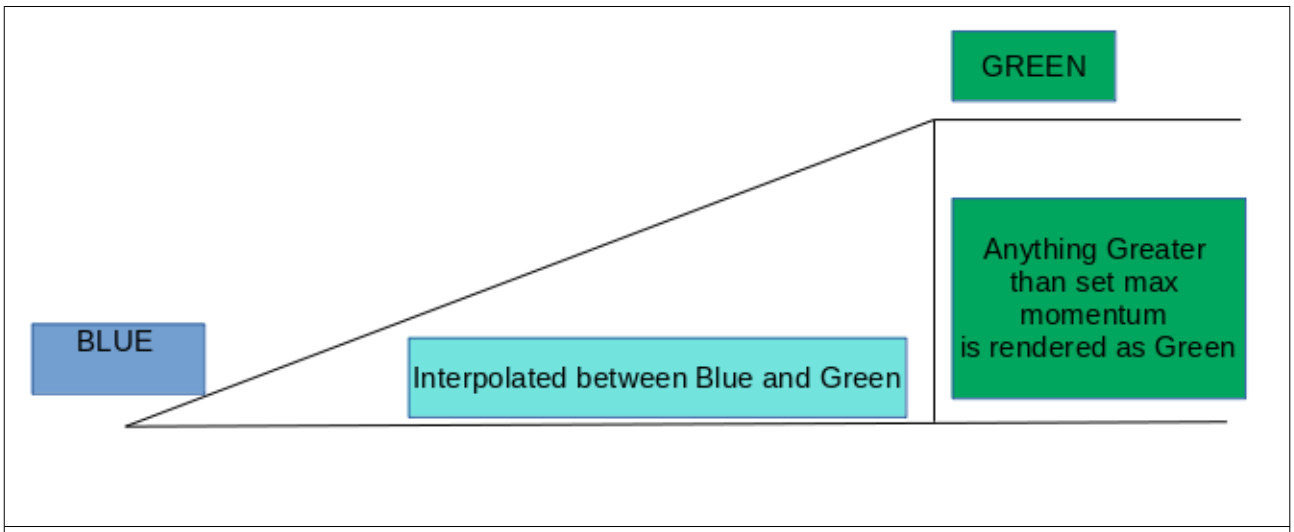

Fig 2: Colour is interpolated between Blue and Green based on its Momentum

The edited lines of code in the function “addWaterColor” in the file “SurfaceAddWaterColor.fs” are the following.

```
void addWaterColor(in vec2 fragCoord,inout vec4 baseColor)
        {
        /* Calculate the water column height above this fragment: */
        //b is the top of the solid ground (sand surface)
        float b=(texture2DRect(bathymetrySampler,vec2(waterLevelTexCoord.x-1.0,waterLevelTexCoord.y-
                1.0)).r+
            texture2DRect(bathymetrySampler,vec2(waterLevelTexCoord.x,waterLevelTexCoord.y-1.0)).r+
            texture2DRect(bathymetrySampler,vec2(waterLevelTexCoord.x-1.0,waterLevelTexCoord.y)).r+
            texture2DRect(bathymetrySampler,waterLevelTexCoord.xy).r)*0.25;
        //This is the vertical thickness of the water layer
        float waterLevel=texture2DRect(quantitySampler,waterLevelTexCoord).r-b;

        /* Check if the surface is under water: */
        if(waterLevel>0.0)
                {
                /* Calculate the water color: */
                // float colorW=max(snoise(vec3(fragCoord*0.05,waterAnimationTime*0.25)),0.0);
                //Simple noise function
                // float colorW=max(turb(vec3(fragCoord*0.05,waterAnimationTime*0.25)),0.0);
                //Turbulence noise

                //Added by Samadrita Karmakar (sam90_karmakar@yahoo.co.in)
                //vx, vy are the x and y momentum components (Divide them by waterLevel to obtain the
                velocity)
                float vx=((texture2DRect(quantitySampler,waterLevelTexCoord).g));
                float vy=((texture2DRect(quantitySampler,waterLevelTexCoord).b));
                float abs_vel=sqrt(vx*vx+vy*vy);
                float max_vel=7.0; //Arbitrary Value
                //Linear Interpolation of Color
                float a1=.075;
                if (abs_vel>max_vel)
                max_vel=abs_vel; //To avoid going over 1.0 for color value
                float a2=(1-a1)/max_vel;
                vec3 setWaterColor;
                setWaterColor.r=.0;
                setWaterColor.g=a1+a2*abs_vel;
                setWaterColor.b=a1+a2*(max_vel-abs_vel);
                //End of Addition by Samadrita Karmakar
                vec3 wn=normalize(vec3(texture2DRect(quantitySampler,vec2(waterLevelTexCoord.x-
                        1.0,waterLevelTexCoord.y)).r-
                  texture2DRect(quantitySampler,vec2(waterLevelTexCoord.x+1.0,waterLevelTexCoord.y)).r,
                  texture2DRect(quantitySampler,vec2(waterLevelTexCoord.x,waterLevelTexCoord.y-1.0)).r-
                  texture2DRect(quantitySampler,vec2(waterLevelTexCoord.x,waterLevelTexCoord.y+1.0)).r,
                                0.25));
```

```
                //Commented  and added out by Samadrita Karmakar
                //float colorW=pow(dot(wn,normalize(vec3(0.075,0.075,1.0))),100.0)*1.0-0.0;
                float colorW=pow(dot(wn,normalize(setWaterColor)),100.0)*1.0-0.0;
                 //Commented  and added out by Samadrita Karmakar
                //vec4 waterColor=vec4(colorW,colorW,1.0,1.0); // Water
                // vec4 waterColor=vec4(1.0-colorW,1.0-colorW*2.0,0.0,1.0); // Lava
                //vec4 waterColor=vec4(0.0,0.0,1.0,1.0); // Blue
                // Addition by Samadrita Karmakar
                vec4 waterColor=vec4(setWaterColor.r,setWaterColor.g,setWaterColor.b,1.0);
                /* Mix the water color with the base surface color based on the water level: */
                baseColor=mix(baseColor,waterColor,min(waterLevel*waterOpacity,1.0));
                }
}
```

4. Results

After running the above edited code, we were successfully able to represent areas with higher vulnerability to erosion, (represented by product of the velocity and the height of water surface above bottom), in green colour and areas with the lower vulnerability to erosion, in blue.

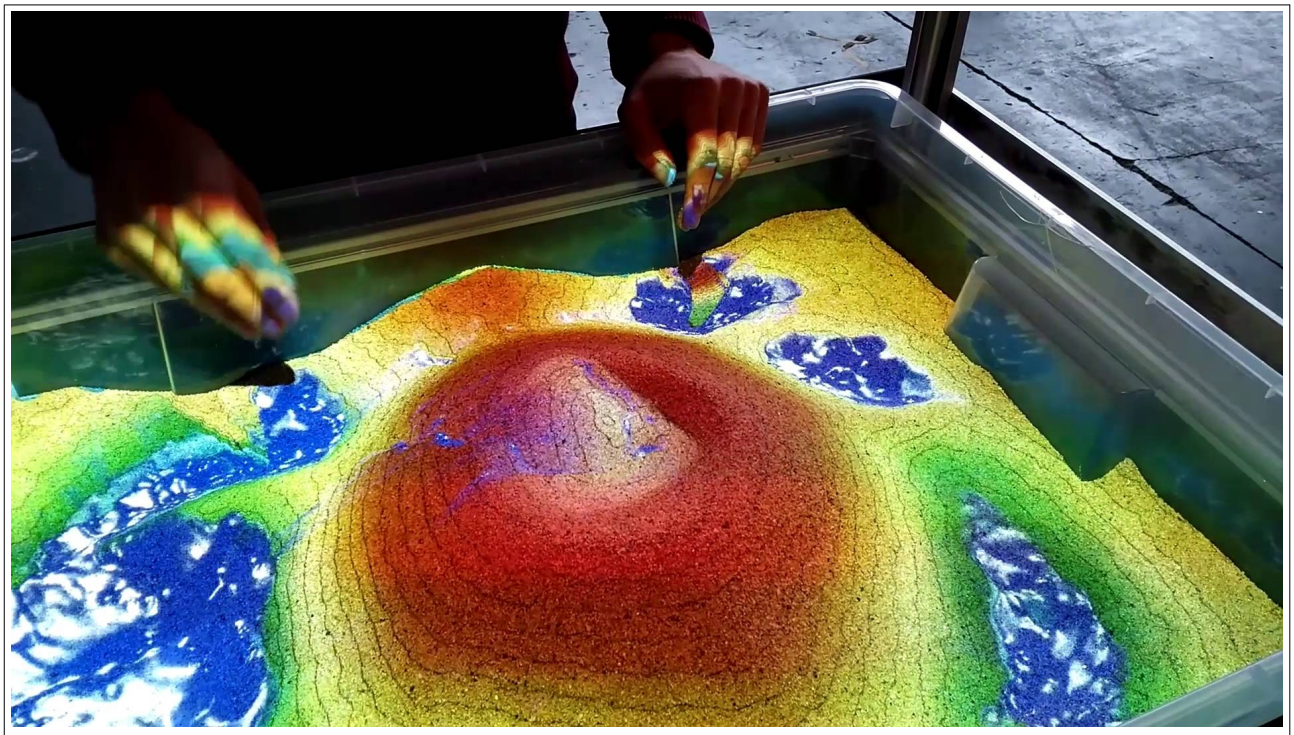

Fig 3: Rendering Before Implementation

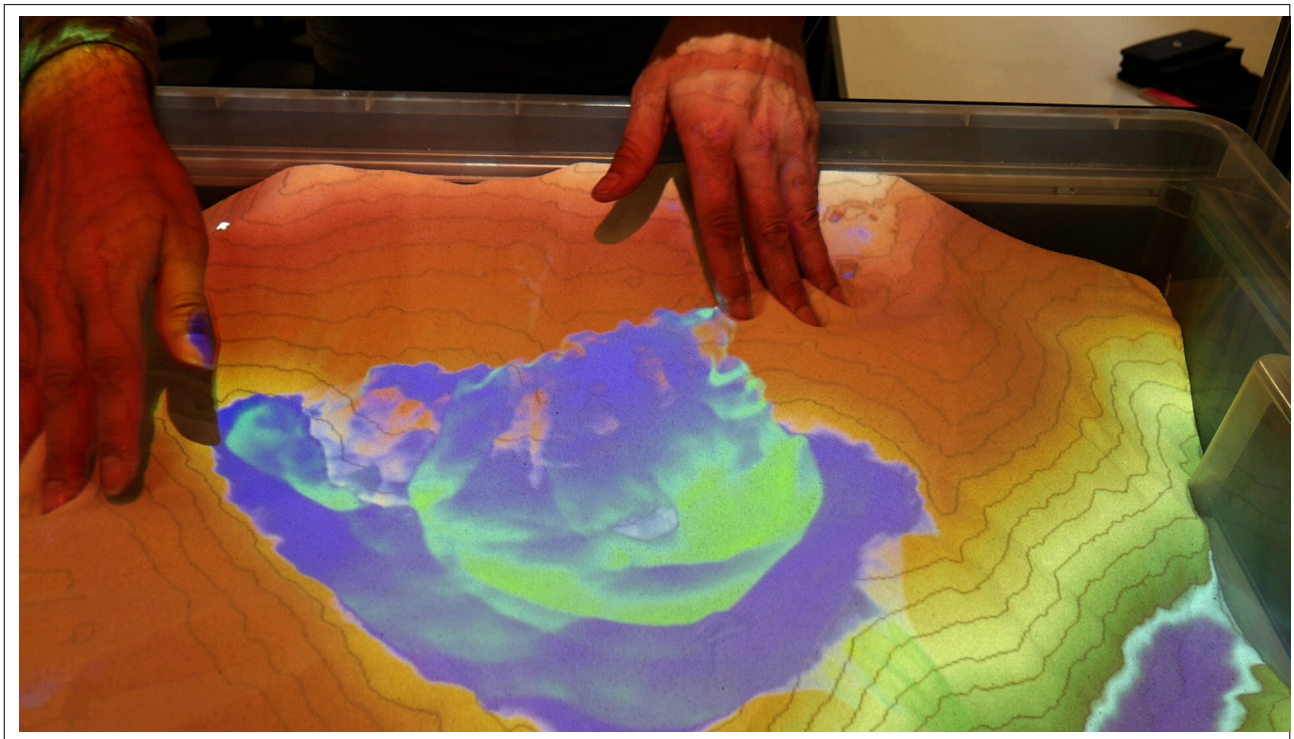

Fig 4a: Rendering After Implementation with Green representing areas of Higher Momentum and Erosion Power

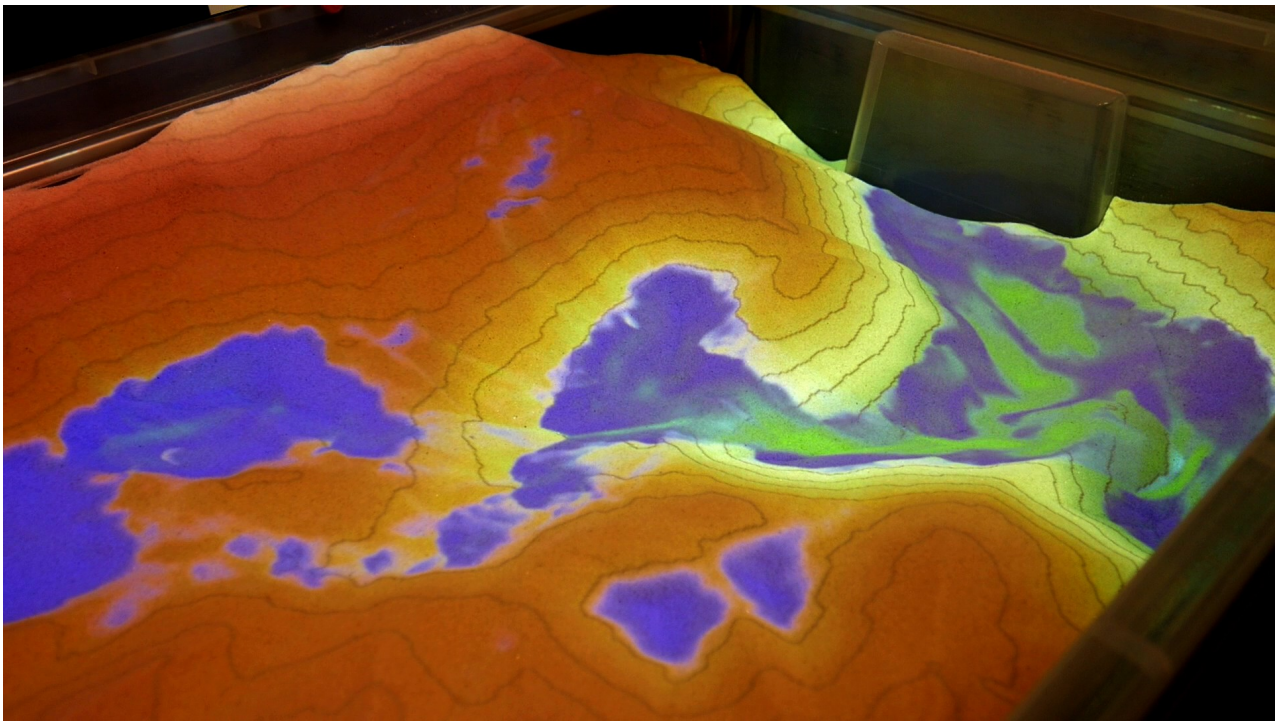

Fig 4b: Rendering After Implementation with Green representing areas of Higher Momentum

The implementation has been simplified to suit the current application but it is similar to erosion rate formulations such as formulations given below for 1D flow regimes.

$$\frac{dz_s}{dt} = -k_e(\tau - \tau_c)^a \quad \text{where} (\tau > \tau_c) \quad \text{and} \quad \tau = \rho g (z_l - z_S) S$$

where,

- $z_l - z_s$ is the water depth
- k_e is the erodability which is an intrinsic property of a given rock lithology
- τ_c being the critical shear stress needed for erosion(a zero value is adopted here)
- τ is the Shear stress at the spillway
- S is the channel slope[6][7]

Thus, the left term represents erosion rate, and $z_l - z_s$ is the water depth.

There are obvious similarities between the above formulation where erosion is dependent on the slope of the channel S , water depth $z_l - z_s$, and the implementation, which is dependent on the product of velocity and height of water surface above bottom.

Due to the above mentioned similarities, the implementation was found adequate enough for visualization purposes.

5. Conclusion

During the internship a free and open source code that does the real-time simulation of water transport based on the scanned surface topology was successfully improved for science outreach purposes.

The Sand-Box along with its underlying code is used in outreach activities to explain to High School students topographic models, freshwater lake ecosystems and earth science processes. The implemented modification allows the instructor to better explain the influence of rivers on the topography through erosion processes.

Finally, keeping the spirit of Free and Open Source Software alive, the source code has been made publicly available on the GitHub page https://github.com/samadritakarmakar/saRndBox_1_6.

6 Source: Garcia-Castellanos & O'Connor, 2018. Scientific Reports (in press). Outburst floods provide erodability estimates consistent with long-term landscape evolution.

7 Source: Garcia-Castellanos, D., Jiménez-Munt, I., **2015**. Topographic evolution and climate aridification during continental collision: insights from computer simulations. **PLOS ONE**, doi:10.1371/journal.pone.0132252