Our GIS is a Game Engine: Bringing Unity to Spatial Simulation of Rockfalls

R.M. Harrap[*1], D.J. Hutchinson[1], Z. Sala[2], M. Ondercin[2,] P.M. Difrancesco[1]

[1]Department of Geological Sciences and Geological Engineering, Queen's University, Kingston, Ontario, Canada K7L3N6
[2]BGC Engineering, Vancouver, B.C.
*Email: Harrap@queensu.ca

Abstract

We employ a game engine – Unity – to create a rockfall simulation that integrates high resolution slope data from LiDAR scanning and photogrammetry, simulates hundreds of rocks moving on a slope, and provides outputs relevant to geological engineering decision support. This approach, in contrast to existing rockfall simulation tools that are specific to limited slope types, is general, extensible, and robust. The use of a game engine as a simulator for surface geological processes represents a novel approach to geocomputation that leverages the physics simulation capability of games and the powerful terrain representation and visualization capabilities of 3D game engines.

Keywords: Game Engine, Geology, Hazards.

1. Introduction

Traditional software tools used to simulate surficial geological processes are custom, typically produced as a deliverable of graduate-level research, and have limited audiences. These tools have no generality: they typically simulate one surface process under specific conditions. Nevertheless, they are essential tools in geological engineering, both for planning new sites and for post-incident analysis.

Video games on the other hand have a wide range of environments they might represent, are produced by expert software developers, and increasingly rely on standardized libraries providing generalized spatio-temporal simulation functionality, as well as standard world construction tools that closely resemble – but vastly exceed many capabilities of – GIS and 3D Visualization tools. These tools and games are now a significant driver both of consumer culture and of computer hardware development, and have budgets to match.

While early computer games aimed simply to provide an (often wildly unrealistic) interactive experience of some sort, modern games increasingly rely on sophisticated feature representation and physics, both because they provide enhanced realism and because a generic 'physics of the world' library can serve for many games. In contrast, a custom solution that avoids simulating processes directly and instead provides a simple proxy of some sort is often not re-usable. There is a direct parallel between the representation issues in GIScience and those in computer games.

The specific domain of interest for this study is the simulation of rockfalls on unstable slopes, where both the dynamic motion of falling rocks and their subsequent impact and position on a road or rail line could damage infrastructure and put humans at risk. While the intent is not to provide a 'game experience,' many of the software subsystems of a game engine (from the world construction tools through the physics subsystem) can be directly used to model a real geological situation and then accurately simulate rockfall behaviour.

By using a game engine, we avoid one-off recreation of all the code required to support a 3D simulation that is not specific to rockfalls. We rely on heavily tested code: hundreds of eyes on, and hundreds of uses for a tool will result in a more reliable – and we believe more accurate – result than a single scientist working alone who is not an expert in software development. The ability to test the same scenario with different code, and run direct comparisons, also allows greater confidence in modeling results.

This paper documents a series of experiments using the Unity game engine to build and evaluate a rockfall simulation tool. It also briefly discusses experiments with other, similar, tools to show the generality of the overall concept of using game industry tools. The resulting tools are both an example of a rather literal geosimulation, and an example of where geosimulation might benefit from an entirely new, robust, and highly functional toolset driven by development budgets orders of magnitude larger than typical scientific research grants.

1.1. What Are Game Engines and Where Did They Come From?

Modern video games are sophisticated blends of user interface, spatial database, simulation, audio, and special effects tools, often with highly tuned peer-to-server or peer-to-peer networking. Gaming has driven the development of graphics technology for several decades, with both increased performance and new, increasingly photorealistic graphics being introduced with each successive generation. As of 2019 big-budget Hollywood effects are achievable in real-time on desktop machines (e.g. Nvidea, 2019).

Early computer games were custom written for a specific platform, and there was little separation between the specific environment represented and the software itself, with the emphasis being on wringing every possible bit of performance out of the hardware (Gregory, 2018).

The development of object-oriented programming methods in the 1970’s and 1980’s, and the shift to modular and interface-based software development that resulted, emphasized re-use of code at all scales of development. With the phenomenal success of Doom and Quake, id Software chose to license their game as an engine for other developers (Abrash, 1997), and the idea of separation between engine and a specific game became commonplace. Since the early 1990’s there have been dozens of commercial game engines released, often tuned to a specific genre of game, with regular updates to keep them in-line with the latest hardware advances.

Over this period the emphasis went from representing a space, to realistic visual representation, to incorporating sophisticated physics both to support representation of the natural world and to support events in-game (like explosions, destruction of materials, and behaviour of fluids). As a result, game engines incorporate sophisticated representation mechanisms for materials, physics engines, databases to store unique and repeating spatial features, methods for procedural generation of entire environments, dialog and interaction AI systems, and physically accurate models for light and sound.

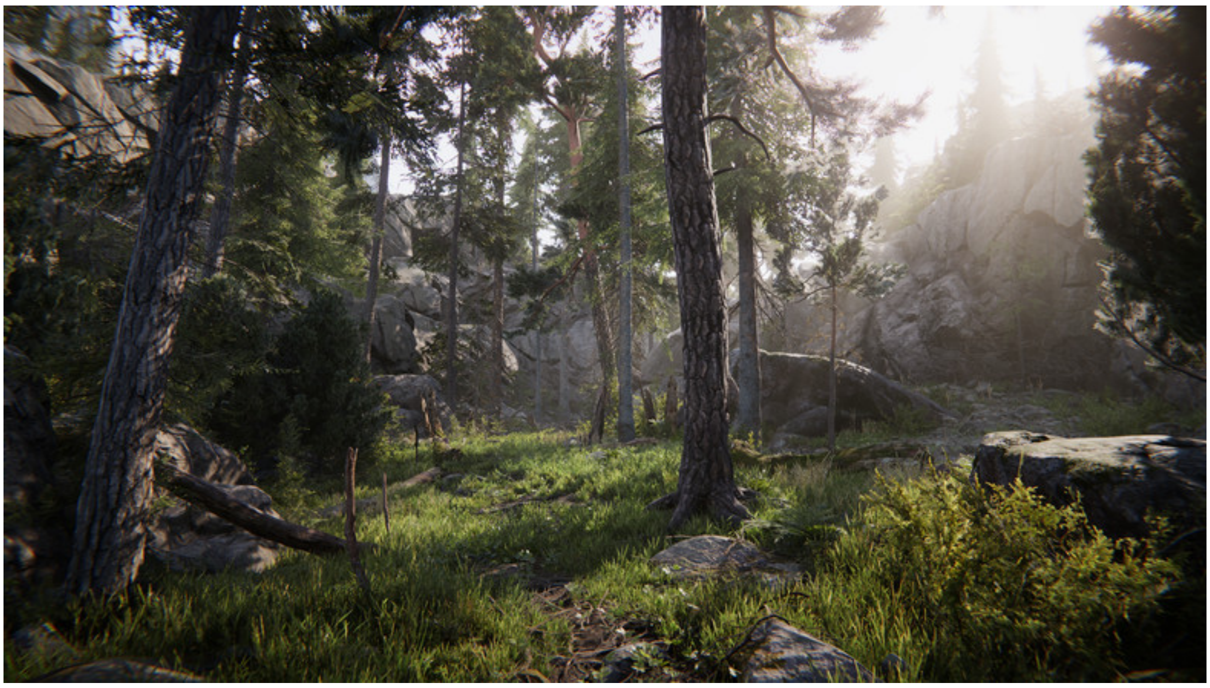

Figure 1: Is it real or is it Unity? A 2018 demonstration game using high resolution graphic assets for a forest environment (Unity Asset Store, 2019 *).

A significant side effect of game engines being reusable has been the involvement of hobbyists making new games or modifying - modding - existing ones. Modding communities are training grounds for aspiring game developers with sophisticated web-based training sites, development teams, and impressive products, for the most part being produced by amateurs.

Although there are many available game engines, we will hereafter focus on discussing two: the Unreal Engine (Unreal, 2019) and Unity (Unity Web, 2019). We will also briefly discuss one tool that operates as a plug-in for game engines – the Houdini toolkit (SideFX, 2019).

1.3 Building Loading, Scripting, Modding

Game engine and authoring environments provide a visualization interface where a world can be constructed, tested, and ultimately packaged and distributed. The essential elements of a game world, all relying on robust code in the provided engine, are lights, objects, sound sources, and special purpose features for special effects such as fires and waterfalls. Generic objects include terrain, buildings, the sky, game characters, and the physical manifestation of features such as lights. Objects can have physics defined – how they interact with other objects upon collision, for example. Game developers have developed a series of knowledge, feature, and process representation frameworks that rival those found in a modern GIS, although with significantly different emphasis. An example of a game development interface, the Unity interface, is shown in Figure 2.

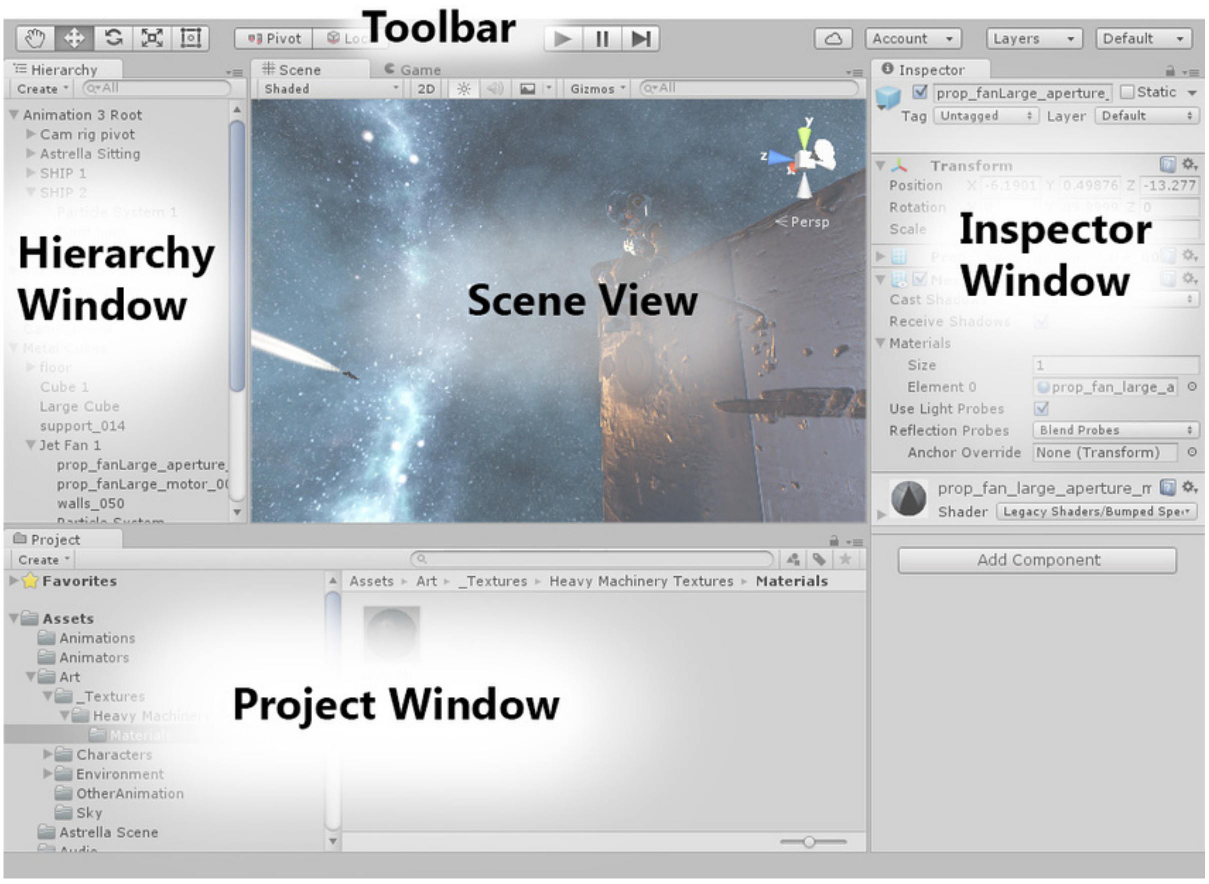

Figure 2: The layout of the Unity development interface (Unity 2019.1 Manual, 2019 *)

The physics engine in a game engine allows a game world to operate as a geosimulation. Each game object can have physical parameters defined and, when the game is run, the behaviour of the object under various forces is simulated. This might include a rock tumbling down a slope, wheels interacting with pavement, or an object responding to an impact from another object. The physics that supports making a good game is, ideally, the physics of the real world.

The game authoring environment supports loading datasets from varied sources, much like a GIS would. Terrain can be imported, models of physical objects can be inserted and varied, and a modular software architecture allows for the insertion of live links (plug-ins) to other software tools with specific capabilities. Details of a game development pipeline are found in, for example, Hallisey (2012).

Object behavior is controlled by scripts attached both to the objects themselves and to the larger environment. Scripts run during game start, for each frame, or when a specific interaction happens. Time is handled both explicitly - events happen in time - and implicitly - some events happen once per screen update.

There are two stand-out game engines widely used at an amateur to professional level at the current time. These are Unity, originally Unity3d to highlight the fact that it emphasized 3d game construction, and Unreal, originally the core of the game Unreal by Epic Games, released in 1998. Both have seen widespread use on projects ranging from independent game design though commercial game projects at major studios. They are also increasingly used to support scientific research, especially projects relying on virtual or augmented reality, since both tools provide excellent support for mobile-device AR/VR development. These environments are extensible, an example being the widely used Houdini toolkit that allows visually scripted effects to be generated either in parallel to or within a game engine.

1.4 Game Engines in Science

There have been significant attempts to use game engines, game development tools, and game communities as a basis of research that falls within the general purview of GIScience in the past. A very brief review of relevant literature follows.

Early examinations of games in science include Lewis et al (2002), Rhyne (2002), and Zyda (2005), Broad discussions of computer games, cartography, and GIScience are provided in Ahlqvist (2011), and of game design principles in Salen et al (2004). Bainbridge (2007) and Ahlqvist et al (2018) discuss the use of analytics to study phenomena related to game communities.

Zyda (2005) draws the link between games and simulation for educational purposes. A detailed case study of the use of game engines, high resolution geospatial data, and scientific concepts to build educational augmented reality experiences is given in Harrap et al. (2012). Serious games – educational games – are discussed in detail in Michael et al (2005). There are also broad parallels between research in using games to simulate social interaction and the GIScience discipline of agent modeling.

The current project, discussed in detail below, is a decision support tool, not a game as such. While it is informed by game and game-like projects in the past, it is closer to geosimulation in that we aim to carry out work that *should* be within the capability of a modern GIS tool but is not. Our direction is quite different. It is more in the spirit of a reply to Gahegan (2018): our GIS is a game engine.

2.0 Using Unity to Simulate Rockfalls

As noted above, the field of geological engineering deals with (among other things) situations where the geological environment directly or indirectly impacts human safety, and infrastructure development or maintenance.

Rockfalls occur when a rock on a slope moves, often energetically. The issue becomes acute when the rock or rocks interact with infrastructure or persons. In the case of rail infrastructure, rockfalls can derail trains immediately, and can destroy or obstruct track causing subsequent derailment. The impact on infrastructure and especially human life can be significant.

Our research group has, in cooperation with Canadian rail companies and the Federal Government, been using an active section of rail line about 200 km NE Vancouver, British Columbia (Figure 3) as a testing area for new ways of detecting, assessing, and ultimately forecasting rockfalls.

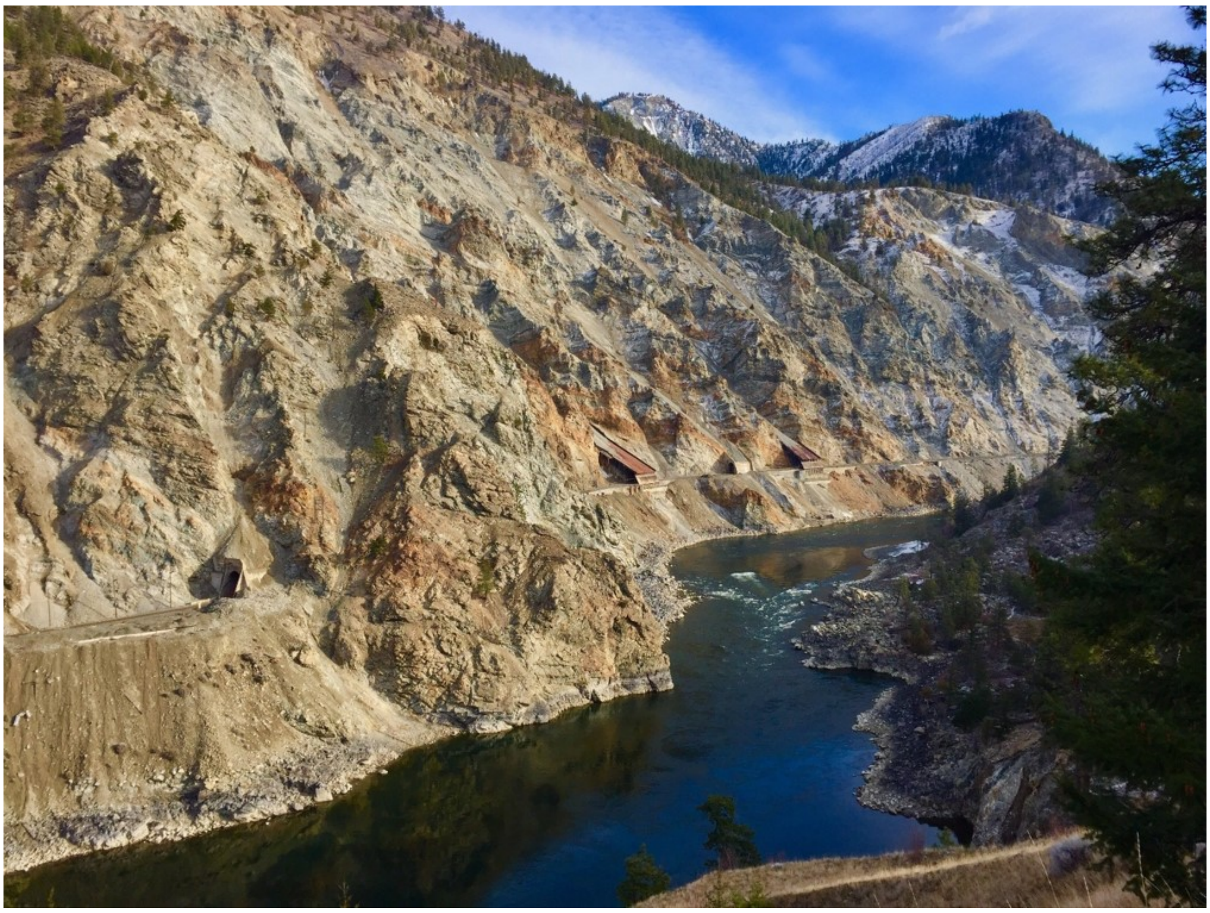

Figure 3: Rail traffic in the Fraser Canyon, B.C. Note the presence of structures to deflect rockfalls over the track and the complex geometry of the slopes generating and deflecting rockfalls. (Photo courtesy D. Bonneau, 2019)

The slope in question (Figure 3) cannot be (safely) directly surveyed, and so we have developed significant protocols using terrestrial and airborne laser scanning and photogrammetry to periodically capture the geometry of the slope. Differencing of subsequent surveys allows an approximate change model to be made, and the change is then attributed to various surface processes including rockfall. Our current models are accurate at the 10cm level over areas many hundreds of meters in length and height, and we have had significant success at both showing correspondence between change detection and known events (Kromer et al, 2018) and in limited forecasting of impending events (van Veen et al, 2018).

Existing rockfall simulation tools suffer from several limitations. Many are two-dimensional: they treat a slope as a profile, and rocks are only simulated in the plane of that profile. Many have limited outputs, and deriving derivative information either for validation or for consequence analysis is difficult. All suffer from being black-boxes, where the internal method used to simulate events ranging from triggering to rock fall itself may not be accessible. Most simulations are based on a lumped mass, where the shape, volume and potential for rock fragmentation during the fall are not considered – all important components affecting the "real" path of a rock down a slope.

As a result, and given the growing capabilities of game engines, we chose to develop a rockfall simulation tool using Unity, to test it against known cases, and to evaluate its performance against other tools. This constituted the MSc theses of two of the authors (Ondercin and Sala).

2.1 Prototype 1: Simulating Rockfalls in Unity

The first implementation of a rockfall model in Unity was by Ondercin (2016). The thesis represented several scenarios in the Fraser Canyon corridor and focused on basic tool development through Unity scripting. Our data sources are at least an order of magnitude more detailed than those normally used in a game, so a significant part of the work was simply working out how to build a model and how to achieve reasonable performance with a high-end gaming computer.

In Unity all objects can be assigned physical parameters (such as mass and bounciness) that control how they act and interact. Time is modelled both in real-world and per-frame update terms. Physics is modeled in real-world time and aspects of interface updating are controlled by the per-frame update to keep the simulation running in real time.

In the first prototype, blocks were placed at the top of a slope, and when the simulation was run the Unity physics subsystem simulated the rocks interacting with the terrain and each other as they moved down the slope.

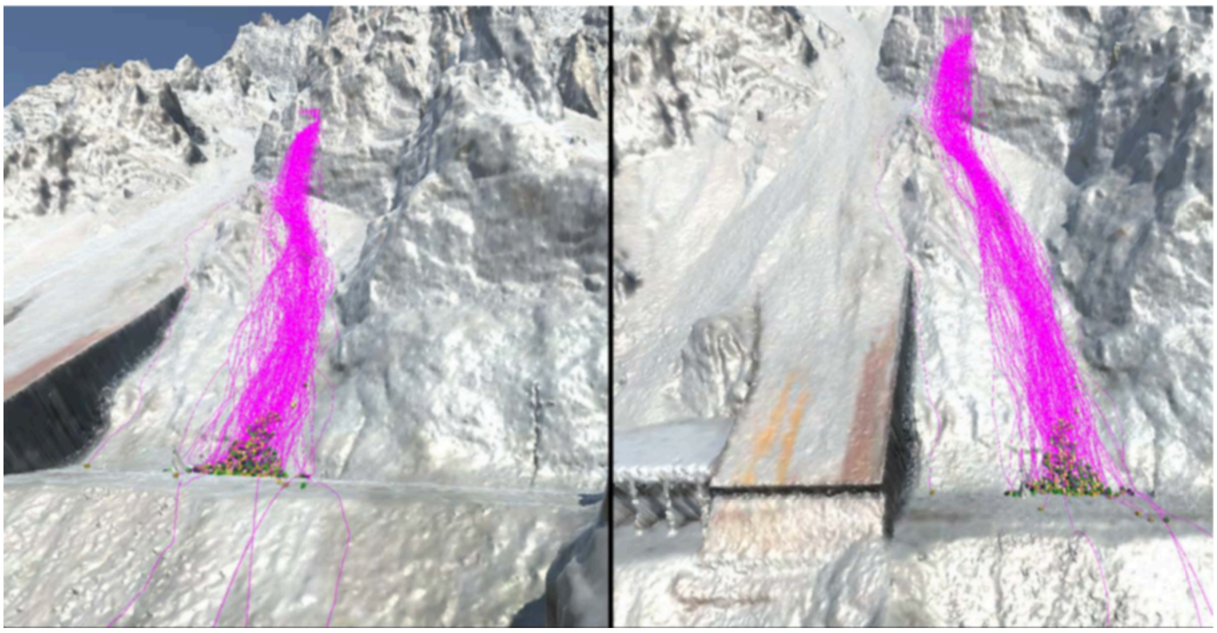

Figure 4: Two views of the same rockfall simulation. The slope is captured by terrestrial LiDAR scanning. This model has 287 blocks of cubic shape and .064 m^3 size. Modified after Ondercin (2016).

In this prototype the blocks have simple geometry, and notably do not fragment.

Since the intent is to produce a tool useable by a geotechnical engineer, the system can output information relevant to slope engineering tasks. For example, a sensor “wall” can be inserted on the slope and each object that passes through that polygon reports all parameters: engineers are often interested in the maximum height of a bouncing block, and the energy of impact, for example, to design a barrier fence or wall (Figure 5). The data collected was passed to a data visualization package (Tableau data analytics software) where statistical analysis and data visualization were carried out.

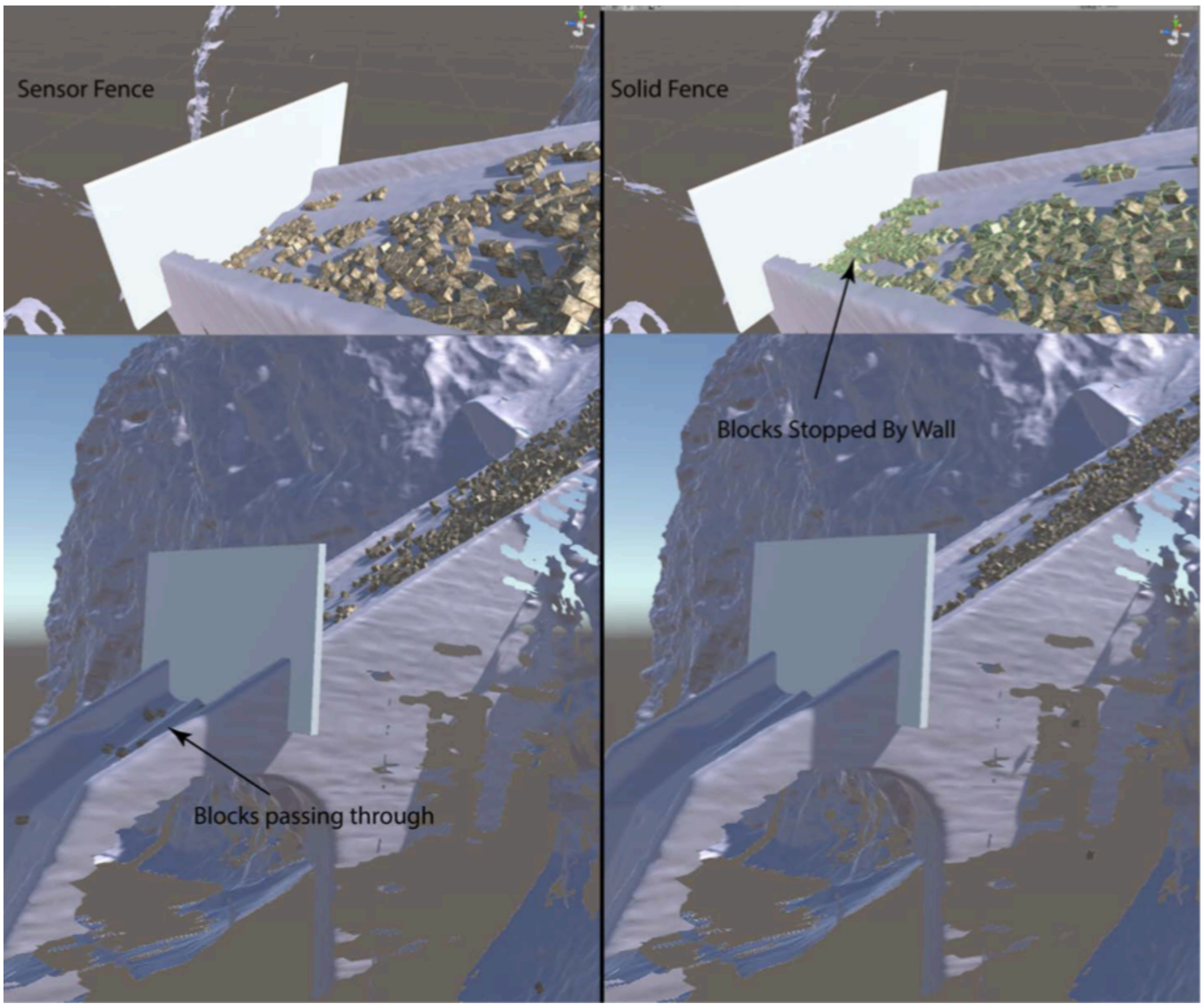

Figure 5: Sensor fences (left figures) and retaining walls (right figures) in the Unity rockfall model. Simulation with 1000 blocks of .008 m^3 (from Ondercin, 2016).

Outstanding issues remaining at the end of this effort included the lack of significant geometric variation between blocks and the lack of precise validation against field cases.

2.2 Prototype 2: Validating Unity Rockfalls

The second phase of development of the Unity rockfall tool was the thesis of Sala (2018). This project focused on block geometry and comparison to existing rockfall case studies. In particular, it relied on several cases where rocks were either deliberately dropped down slopes (Ushiro et al, 2006; Vick, 2015; Volkwein et al, 2018), or where sufficient records existed in the study site database to precisely determine the source volume and geometry, impact positions, and deposition distribution and fragment sizes. This study is documented in detail by Sala (2018) and Sala et al. (in press).

The first issue addressed was block fragmentation. Given the available slope data, we often have superb pre-fall geometry for a zone that fails, but only in rare cases see how a falling mass subdivides both initially (as it is triggered) or subsequently (as it bounces or impacts other falling masses). By making geologically reasonable assumptions about fragmentation, a model was constructed where a block fragments during transport.

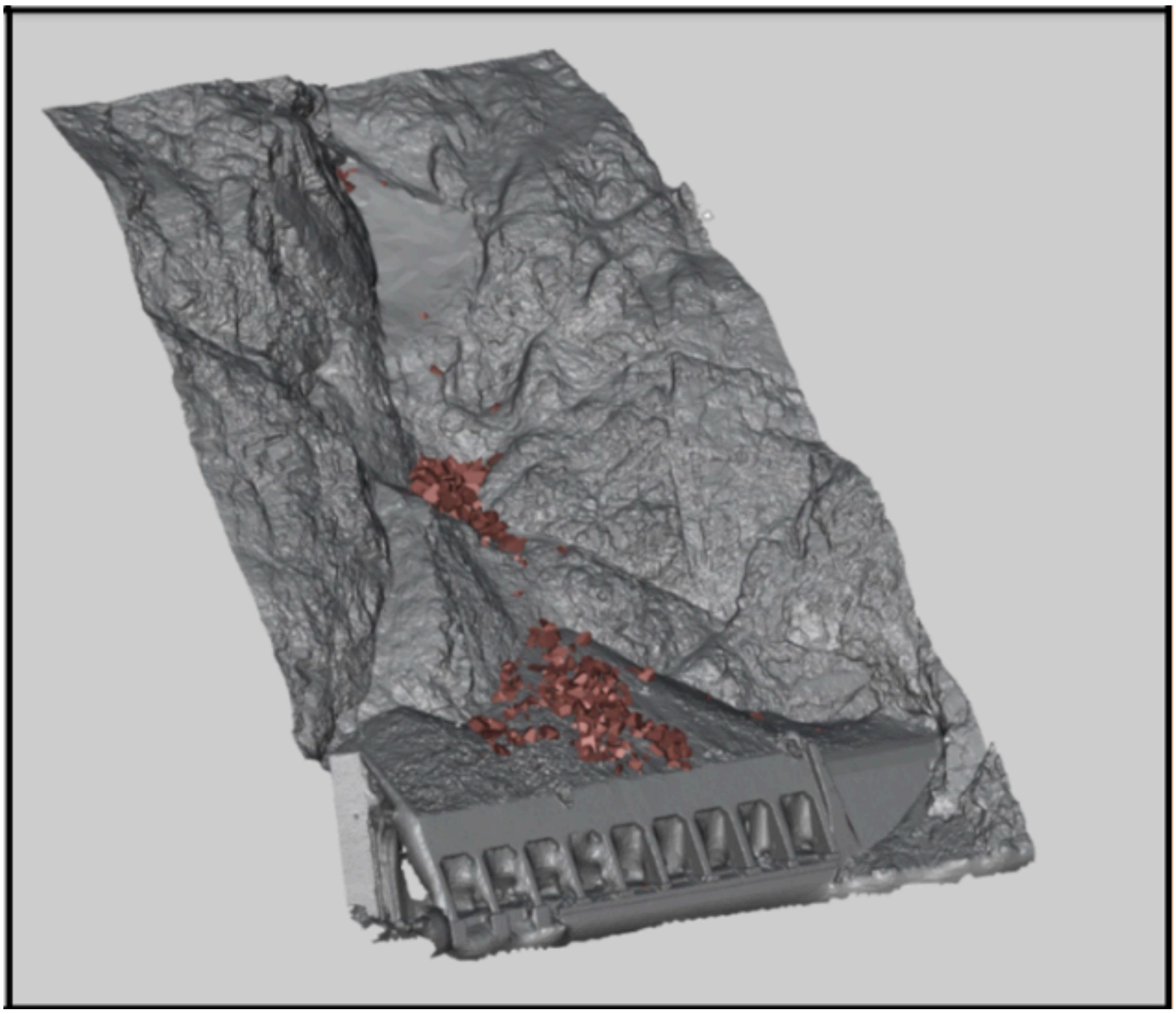

Figure 6: 1000 fragment rockfall simulation using discrete and variable fragment shapes. Total volume is 100 m^3 (from Sala et al. in press).

Several case studies were then recreated mirroring the case studies from the literature, to determine if the Unity model provides impact points and energetics similar to what was observed in the real world. The coherence between the models and the observational data was excellent, as documented in Sala (2018).

2.3 Experiments with Other Tools

Briefly, we have experimented with environment creation and scripting in both the Unreal Engine and in the Houdini effects package introduced above. We highlight work in progress by one of the authors (Difrancesco) that examines Houdini use for the same slopes and same environmental conditions. Houdini uses a procedural node-based workflow (Figure 7) and is thus programmed visually. A breakable object set was defined and released on the slope. It was rendered both with a photorealistic render option (24 hours on a modern desktop PC) and with the internal quick render option as a ‘Flipbook’ (1 hour). Fragmentation results were superb (Figure 8) but, unlike Unity, the system is clearly not real-time.

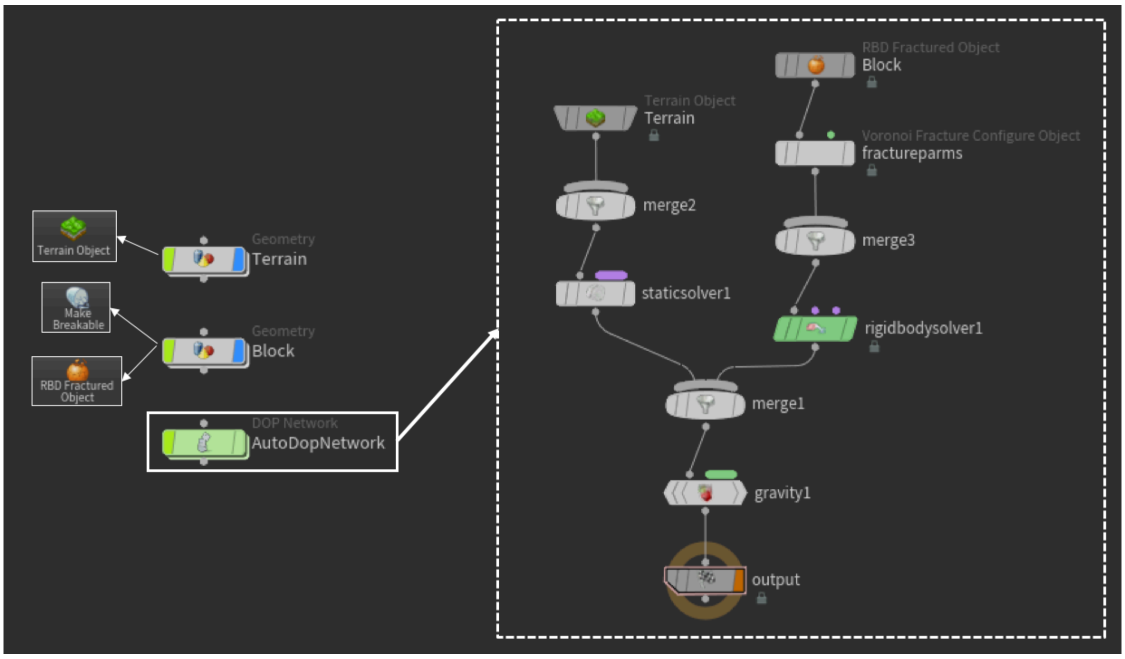

Figure 7: Hierarchical visual programming of a rockfall simulation in Houdini.

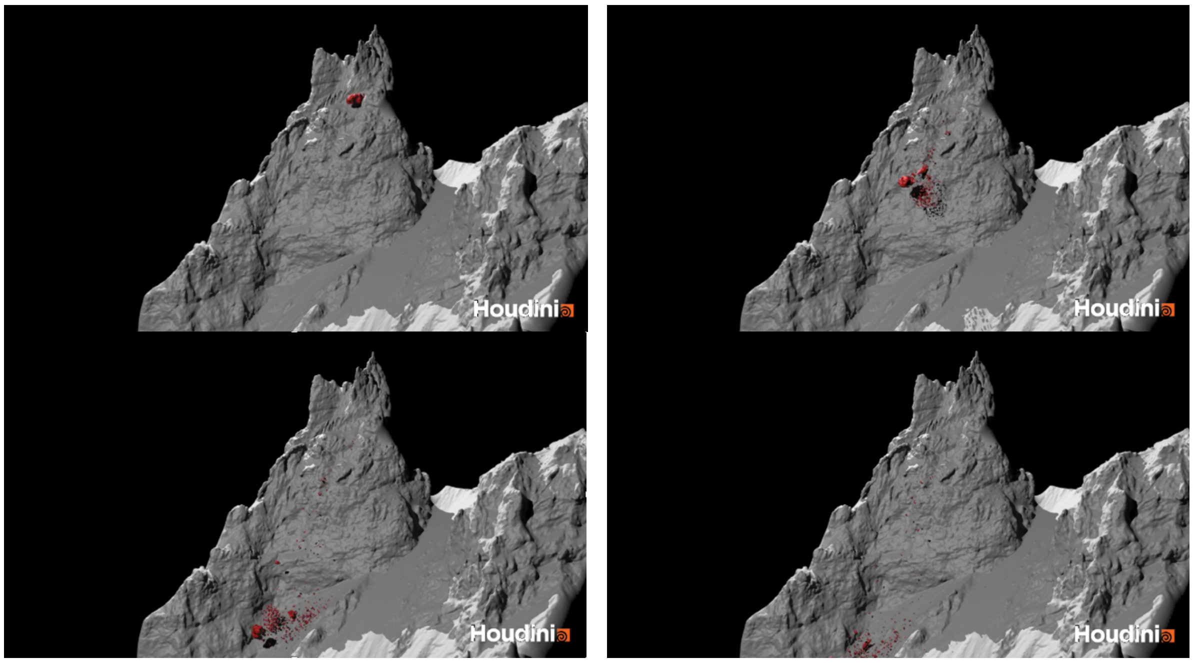

Figure 8: Extract from simulation of rockfall with material-based fragmentation.

Work in both Unreal and Houdini to compare and contrast with Unity continues.

3.0 Discussion and Future Work

Our ongoing attempts to model rockfalls represent a specific case of geological geosimulation. The novelty of our approach is in that we were quickly able to produce results that appear to be more realistic by relying on the software development environment provided by Unity, and that this then provided several distinct advantages:

1. Using Unity physics engine component allowed us to avoid long periods of development.

2. The Unity tool ecosystem allowed rapid world and feature creation.

3. Unity supports add-ons such as agent-based simulation, allowing follow-on studies that move from hazard to risk analysis.

4. Our code – the scripts needed to add the rockfall simulation to Unity – can be freely distributed, and Unity is widely available at no cost for academic use.

There were a number of limitations to the studies done to date. We are working to address these as development continues. As noted, we are also making similar, limited experiments with other tools.

While we can replicate real world tests, testing against different physics models – the physics engine in Unity is a closed subsystem – has been a problem. As of 2019, Unity supports multiple physics engines allowing internal validation. Note that the community studying rockfalls do not have enough calibration data sets at the moment to come up with a recommended, verified set of physics parameters; this both limits validation of models and our ability to directly compare results from different tools.

While we are now simulating disparate rock fragment shapes and sizes, we are not directly simulating rock mechanics. This is both a scientific and a computing performance problem. The performance issue will be addressed through GPU-based computing. The scientific problem requires a number of advances: better understanding of the local geology, better understanding of the collision processes, and a shift from approximations for particle interactions to more robust (but computation-heavy) representations. The interaction issue is being addressed by studies where rocks are instrumented and observed over many rockfall events. The slope geology problem is being addressed by better remote sensing approaches, but ultimately geological engineering is always limited by both the scale of observation of the Earth and the inherent variability of geological materials. A completely accurate representation of surface and subsurface geology is a game we cannot win.

A playlist of example videos is referenced in the paper notes below.

4.0 Conclusions

This initiative, to represent and simulate rockfall processes in Unity, responds to some of the criticism of GIScience posed by Gahegan (2018): that there is a need to study temporal phenomena, that a wider view of representation is needed, and that there is a need to have a larger community active in software development. We leverage a spatiotemporal simulation engine – a game engine – with significant physical modeling capabilities to develop and test models of rockfalls. Our models are open source and run on the freely available Unity engine. Related, preliminary models run in Houdini and Unreal. We successfully replicate real world cases and have the ability (unlike commercial rockfall tools) of easy modification, linkage to other software packages, and ease of transfer of terrain data from other products.

This approach can be generalized to a wide variety of cases where physical process simulation is desired, and has direct links to the application areas of civil engineering and infrastructure, where simulating infrastructure proposals under different traffic and natural hazard scenarios is required.

This approach has the further advantage, echoing Gahegan (2018), that there is a huge resource of young scientists schooled in game development, eager to apply those skills, and looking to make a mark. The skills these students acquired modding games are directly transferable to projects like ours.

The budgets of game engine providers and game studios dwarf the budgets of individual researchers, or even entire fields of research at an international level. Since advances in realistic physical simulation of processes makes these tools better, and consequently result in better and cheaper games, there is a significant opportunity for cooperative tool development between the game industry and researchers in GIScience.

5.0 Acknowledgements

A number of students have been involved in the studies of the White Canyon in the Thompson River , and their work enabled the construction of the models used here. Thanks to D. Gauthier, R. Kromer, M. van Veen, M. Lato, and D. Bonneau for their work. Thanks to NSERC, CN Rail and Canadian Pacific for their funding of this Collaborative Research and Development project.

6.0 Notes

A playlist (not searchable, but available via this link) has been posted at:
https://www.youtube.com/playlist?list=PLJzaXhGZdpBbyyZR_OhGHc5OoV47tCWzk

images indicated with an "" used with permission of Unity Technologies; "Unity" and Unity logos are registered trademarks and trademarks of Unity Technologies or its affiliates in the US and elsewhere. Neither this work nor its author is affiliated with, or endorsed or sponsored by, Unity Technologies or its affiliates.

7.0 References

Abrash, M. 1997 Quake's Game Engine: The Big Picture. *Dr. Dobb's Journal*, spring.

Ahlqvist, O. 2011 Converging Themes in Cartography and Computer Games. *Cartography and Geographic Information Science,* 38(3), pp. 278-285.

Ahlqvist, O., Khodke, N., and Ramnath, R. 2018 GeoGame analytics – A cyber-enabled petri-dish for geographic modeling and simulation. *Computers, Environment, and Urban Systems.* 67(1), pp. 1-8.

Bainbridge, W.S. 2007 The Scientific Research Potential of Virtual Worlds. *Science*, 317(5837), pp. 472-476.

Gahegan, M 2018 Our GIS is too small. *The Canadian Geographer,* 62(1), 15-26.

Gregory, J. 2018 *Game Engine Architecture 3ed*. Boca Raton: CRC Press.

Hallisey, B. 2012 Building a Virtual World: The Pipeline and the Process. *IEEE Computer*, 45(12), pp. 90-92.

Harrap, R., Daniel, S., Power, M., Pearce, J. and Hedley, N. 2012 Design and Implementation of Mobile Educational Games: Networks for Innovation. In: Chrisman, N. & Wachowicz, M. (Eds.), *The Added Value of Scientific Networking: Perspectives from the GEOIDE Network Members 1998-2012.* Quebec City: GEOIDE Inc. pp. 157-187.

Kromer, R.A., Rowe, E., Hutchinson, J., Lato, M., and Abellán, A. 2018. Rockfall risk management using a pre-failure deformation database. *Landslides*, 15(5), pp. 847–858.

Lewis, M., Jacobson, J. 2002 Game Engines in Scientific Research. *Communications of the ACM*, 41(1), pp. 27-31.

Michael, D.R. and Chen, S.L. 2005 *Serious Games: Games that Educate, Train, and Inform*. Boston, Mass: Thomson Course Technology.

NVidea 2019 *NVidea RTX Ray Tracing.* [Online]. [Accessed May 2, 2019]. Available from: https://developer.nvidia.com/rtx/raytracing

Ondercin, M. 2016 *An Exploration of Rockfall Modelling Through Game Engines.* MSc thesis, Queen's University at Kingston, 200pp.

Rhyne, T-M. 2002 Computer Games and Scientific Visualization. *Communications of the ACM*, 45(7), pp. 40-44.

Sala, Z. 2018 *Game-Engine Based Rockfall Modelling: Testing and Application of a New Rockfall Simulation Tool.* MSc Thesis, Queen's University, 206pp.

Sala, Z, Hutchinson, D.J., and Harrap, R.H. [Forthcoming]. Simulation of Fragmental Rockfalls Detected Using Terrestrial Laser Scans from Rock Slopes in South-Central British Columbia, Canada. *Natural Hazards and Earth Systems Sciences.*

Salen, K., and Zimmerman, E. 2003 *Rules of Play: Game Design Fundamentals.* Boston: MIT Press.

SideFX 2019 What is Houdini? [Accessed June 15, 2019]. Available from: https://www.sidefx.com/products/houdini/

Unity Asset Store 2019 *Book of the Dead Environment.* [Online]. [Image accessed May 1, 2019]. Available from: https://assetstore.unity.com/packages/essentials/tutorial-projects/book-of-the-dead-environment-121175

Unity Web 2019 *Unity For All.* [Online]. [Accessed May 1, 2019]. Available from: https://www.unity.com

Unity 2019.1 Manual 2019 *Learning the Interface Module,* Unity 2019.1 Manual. [Online]. [Image accessed May 1, 2019]. Available from: https://docs.unity3d.com/Manual/LearningtheInterface.html

Unreal 2019 *What is Unreal.* [Online]. [Accessed May 1, 2019]. Available from https://www.unrealengine.com

Ushiro, T., Kusumoto, M., Shinohara, S., and Kinoshita, K. 2006. An experimental study related to rock fall movement mechanism (In Japanese). *Journal of Japanese Society of Civil Engineers*, 62(2), pp. 377-386

van Veen, M., Hutchinson, D.J., Bonneau, D.A., Sala, Z., Ondercin, M., and Lato, M. 2018. Combining temporal 3-D remote sensing data with spatial rockfall simulations for improved understanding of hazardous slopes within rail corridors. *Natural Hazards Earth System Sciences*, 18(8), pp. 2295-2308.

Vick, L.M. 2015. *Evaluation of field data and 3D modelling for rockfall hazard assessment.* Phd thesis, Department of Geological Sciences, University of Canterbury, Christchurch, Canterbury, New Zealand.

Volkwein, A., Brügger, L., Gees, F., Gerber, W., Krummenacher, B., Kummer, P., Lardon, J., and Sutter, T. 2018. Repetitive Rockfall Trajectory Testing*. Geosciences*, 8(3): 88.

Zyda, M. 2005 From Visual Simulation To Virtual Reality To Games*. IEEE Computer*, 38(9), pp. 24-32.