PointCraft: Harnessing Players' FPS Skills to Interactively Trace Point Clouds in 3D

Kathleen Tuite
University of Washington
ktuite@cs.washington.edu

Rahul Banerjee
University of Washington
banerjee@cs.washington.edu

Noah Snavely
Cornell University
cornell@cs.cornell.edu

Jovan Popović
Adobe Systems
jovan@adobe.com

Zoran Popović
University of Washington
zoran@cs.washington.edu

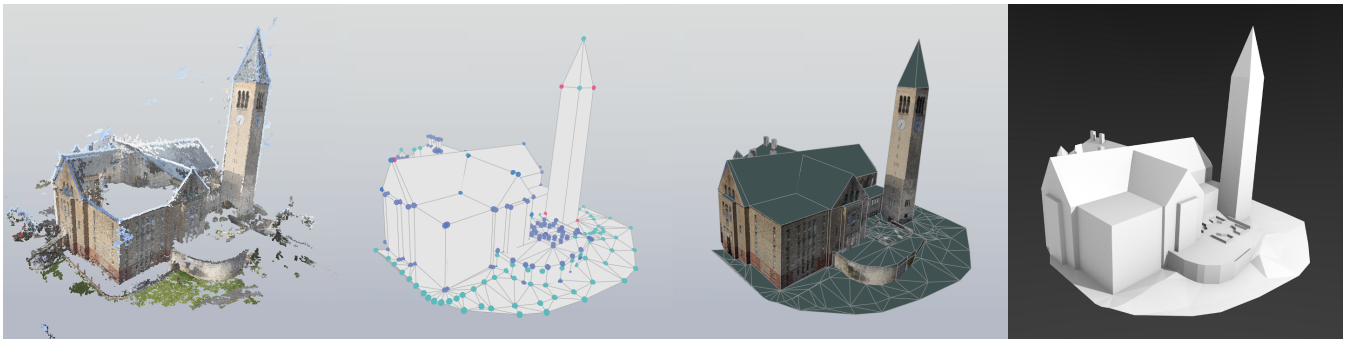


Figure 1: Point cloud, polygons and pellets, model with texture, and rendered geometry shown for a building generated from photos.

ABSTRACT

We introduce PointCraft, a 3D modeling tool to complement the crowdsourced photography game PhotoCity, which is heavily inspired by the voxel world-building game Minecraft. The game with a purpose PhotoCity produces noisy, incomplete point clouds from thousands of player photographs, and automatic methods do not currently work to transform these point clouds into clean, concise, usable geometry. PointCraft is a tool to harness the power of existing players to model these point clouds with human interaction. Drawing on first-person navigation skills many game players have already developed, and specifically on the first-person modeling skills of Minecraft, PointCraft provides a new opportunity for players to contribute to the overarching goal of reconstructing a virtual model of the world. Our primary contribution is a UI for interactively tracing point clouds in 3D, but we also wish to emphasize the idea of adapting familiar game controls to new, serious purposes where human intervention is incredibly valuable. Furthermore, we believe in the importance of attracting a wide audience and where many players with different interests and skills can collaborate on the same goals.

Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), June 22-25, 2015, Pacific Grove, CA, USA.
ISBN 978-0-9913982-4-9.

Keywords

3D modeling, tracing, gwaps, games, crowdsourcing

Categories and Subject Descriptors

H.8.0 [**Information Systems**]: General – Games; I.3.6 [**Computer Graphics**]: Methodology and Techniques

1. INTRODUCTION

We introduce PointCraft, a game-inspired tool for interactively tracing point clouds in 3D. PointCraft is designed for working on point clouds of buildings (left image in Figure 1, Figures 3, 4, 11) generated automatically from thousands of photographs (from different angles) captured by players of the game PhotoCity [26]. PhotoCity's overarching goal was to reconstruct 3D models of the whole world, but there was no way to get clean, concise, useful geometry from the noisy and incomplete point clouds produced. Some automatic methods like Poisson Reconstruction [12] make no assumptions about the fact that we are reconstructing urban architectural scenes (producing blobby artifacts as shown in Figure 2) while others [11, 16] are too constrained to work with the variety of models found in PhotoCity. The point clouds themselves are easily understandable by humans, and through PhotoCity, there is already a source of players who are interested and invested in the photography side of the problem. Our goal is to provide a new

tool for existing PhotoCity players to complete the geometric modeling process, but make it accessible to new players familiar with traditional videogames as well, so that more people may creatively contribute to the modeling process.

To understand PointCraft's contribution, we first need to describe point clouds and where they come from. Point clouds are collections of 3D points and may be obtained from a variety of sources, such as LiDAR scanners, depth-sensing cameras (e.g. Microsoft Kinect, Leap Motion) or image-based techniques such as Structure from Motion, which was used by PhotoCity. Unlike lines or polygons, points are zero-dimensional geometric objects, and therefore hard to visualize. Depicting each point as a 3D sphere (for instance) causes nearby points to coalesce into a blobby mess. Rendering them as individual pixels does not obscure more distant points, the way that a filled polygon in the foreground occludes other primitives behind it.

A traditional 3D modeling tool such as Maya is too complex and cumbersome to expect an average PhotoCity player to use. Furthermore, point clouds pose their own challenges for interpreting, navigating, and manipulating. First, it can be difficult to interpret depth when distant regions of the point cloud show through close regions. Also, it can be disorienting to navigate large models (e.g. reconstructions of urban scenes) using camera controls like the arcball, traditionally designed for viewing smaller objects. Finally, it can be frustrating to interact with and try to select points on a point cloud when there is no explicit surface to point a cursor at.

In order to develop a modeling environment that would seem familiar and easy to use, we took inspiration from the first-person voxel world-building game Minecraft, popular with millions of people around the world [21]. We found that providing users with natural walking (and flying) controls allowed them to navigate large point clouds in a more familiar way. Such motion also provided parallax, a depth cue otherwise missing from the point cloud, which helped the user understand the scene.

Similar to Minecraft's use of voxels as its modeling primitives, PointCraft uses a spherical *pellet* as its modeling primitive. The point cloud becomes not just a visual guide to the modeler, but a *virtually tangible* guide, something the modeler can grab on to and attach pellets to. As a result, modeling the 3D shape becuomes as easy as *tracing* the 3D point cloud, in the same way someone would trace a picture in 2D.

PointCraft not only makes the interactive tracing and modeling of point clouds possible, it addresses the difficulty of working with point clouds by combining the interaction and navigation into a single mode. This first-person modeling paradigm is an excellent fit for our domain. Our primary contribution is the PointCraft UI itself and demonstration of its use to trace point clouds from PhotoCity. By building this new tool, we are closer to achieving the original goal of PhotoCity, to collaboratively and collectively reconstruct a 3D model of the world. We also wish to emphasize the idea of adapting familiar game controls to new, serious purposes where human intervention is incredibly valuable for solving problems that automatic techniques struggle with. Additionally, by extending an existing game with a purpose like PhotoCity with a new, related tool, we hope to widen the potential player audience and allow many different ways for players with different skills and interests to contribute to common goals.

2. RELATED WORK

Games with a purpose leverage human common sense, intuition, and/or problem solving abilities to solve real world problems. In some cases, the player simply contributes labels, as in von Ahn's ESP Game [28], but in other games, players use custom tools to control deep, domain-specific interactions. In the game Foldit, players manipulate virtual proteins in 3D in order to predict their folding structures [7], invent their own folding algorithms [6], and even design and restructure proteins [9]. Likewise, players of Eterna are discovering new rules for RNA structure design [15]. Our tool PointCraft has a similar premise to Foldit, in that there is a 3D problem that computers struggle with (automatically reconstructing noisy point clouds), but with the right tools, humans can use their own intuition and judgements to fill in the details and accomplish the task.

Before we talk about the ways humans can be involved in the task of modeling geometry from point clouds, we must discuss the history of automatically generating meshes from unstructured point clouds. Over the last two decades, automatic methods have successively improved [12, 1, 3, 2, 14], but they rely on data to be both accurate and reasonably complete without holes. Real world point clouds, especially those generated from photographs captured by multiple users with different cameras, usually violate both these assumptions. Using automatic methods on such point clouds results in reconstructions that look like Figure 2.

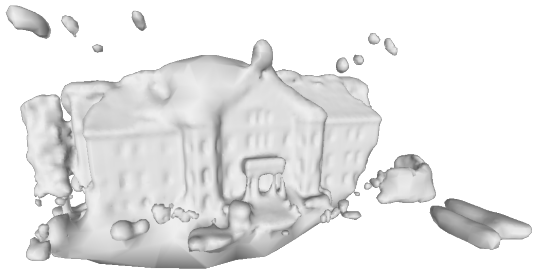


Figure 2: Automatic Poisson Reconstruction of Lewis Hall point cloud. Because the algorithm is unaware of architectural motifs and context for the scene, it smoothes out sharp edges, creates floating orbs out of noise and loose points, and makes unrealistic approximations where data is missing.

The type of model we desire is a simplified polygonal model, suitable for importing into Google Earth. To borrow a phrase from Chauve et al. [4], we would like "concise and idealized" geometry. Several automatic methods exist for generating piecewise-planar models from point clouds. Some use RANSAC [23] to generate candidates, while Chauve et al. use a region-growing approach. The approaches of Nan [17] and Li [16] incorporate a-priori knowledge about the structure of the input data (architectural buildings with repeated structure and small objects composed of basic primitives, respectively) which precludes their use on other types of point clouds. These automatic methods could work for portions of PhotoCity point clouds known to be highly structured and repetitive, but in general, the models we looked at were too complex and heterogeneous, containing many different types of architecture as well as plants and non-architectural objects.

Some systems exist which allow interactive modeling from point clouds, though they are too limited or too specific for our purposes. PointShop3D [30] was the first publicly-available software that allowed users to interact with the point cloud data directly. Pauly et al. [20] extended it, allowing the user to directly manipulate a hybrid surface model inside their software. Weyrich et al. [29] added tools for touching up the severe scanning artifacts present in many point clouds. Weyrich's tools in particular are designed for mostly complete point clouds and therefore not very effective when large swaths of geometry are absent. For instance, in Figure 1, the point cloud is missing its roof, which we were able to fill in using PointCraft. The work of Colburn et al. [5] lets users place walls of a house using a point cloud as a visual guide. Since the end goal is *re*modeling houses, their system does not support the creation of arbitrary geometry, while PointCraft does.

Figure 3: Two screenshots of the PointCraft UI showing a user defining a polygon.

Since our point clouds are generated from images, we now discuss similar image-based modeling approaches. These techniques involve tracing on images or video frames and have proven to be quite powerful [8, 18, 24, 27] and capable of producing clean, yet detailed models. Debevec et al. [8] exploited symmetry in architectural structures to recover geometry. More recently, Sinha et al. [24] showed how vanishing lines (pre-computed from the images) can be used as a visual guide for letting the user draw geometry on images. However, deciding which photos to present to the user from among potentially thousands of images is a non-trivial task. Further, using an image as a viewpoint precludes modeling from other vantage points (such as overhead views or extreme close ups).

There have been attempts to simplify 3D modeling via sketch-based interfaces [13, 19]. These systems leverage a more "natural" interface (typically a stylus on a touch-sensitive tablet) thereby enabling any user to create 3D geometry. However, by working on a 2D surface, the user does not have direct control over the created 3D geometry. Schmidt [22] added image-space linear 3D scaffolding to act as visual constraints, thereby giving the user more precise control over the 3D authoring process. PointCraft follows a similar approach, letting the user create 3D scaffolding to guide construction of primitives and/or more scaffolding.

Interactive 3D modeling software has traditionally been dominated by powerful, but hard-to-use high-end software such as Maya, SolidWorks, and CATIA that takes weeks to learn and months, if not years, to master. Our approach with PointCraft was to make the interface as simple and familiar (especially to people who have played Minecraft and other first-person games) as possible, allowing non-expert users to quickly become productive with it.

3. USER INTERFACE

PointCraft is heavily inspired by Minecraft, "a game about placing blocks to build anything you can imagine." In Minecraft, players can build anything they want, as long as they can represent it using blocks. With PointCraft, we wanted to give users the ability to easily model real-world scenes, provided they're available as point clouds.

In designing PointCraft, we borrowed the following ideas from Minecraft:

- First-person navigation (using the mouse to look around and keys to move forward/backward/left/right/up/down) which in our case provides motion depth cues and a natural 'walking' metaphor for navigating visually complex point clouds.
- Combined navigation and modeling, allowing the user to move around their work as they build, instead of explicitly switching modes.
- Tools based on a simple primitive (in our case, a spherical pellet instead of a Minecraft block) that are easy to explore and experiment with.
- A minimal head-up display (Figure 3) showing only the current tool, the user's last action, and the tool palette. Users change tools with the number keys (a convention in shooter games), limiting the tools directly accessible at any point in time to just ten.
- An extended tool inventory from which users can customize their on-screen palette. (See Figure 6).

3.1 First-Person Navigation Controls

Our design goal with PointCraft is to allow videogame players to interactively trace point clouds, but navigating and interpreting point clouds can be confusing and disorienting. Multiple layers of 'surfaces' can be visible at the same time, especially when the point cloud is imperfect and incomplete. Additionally, cloud data without normals or with noisy normals lack cues about orientation generally provided through shading on solid surfaces. Luckily, motion (of the viewpoint) provides us with the depth and orientation cues otherwise missing from static images of point clouds.

Since 3D games have already trained players how to navigate large outdoor scenes via first-person controls, it is a natural design choice to exploit this in a game with a purpose that requires navigating 3D environments. Many of our point clouds are of buildings and large outdoor scenes featuring multiple buildings, and we have oriented them to give them a sensible up-direction. Users can walk forward and backward and strafe left and right with the conventional keys while changing the direction they are looking with the mouse. Walking forward always moves the user horizontally in the viewing direction. This keeps the user attached to an invisible 'ground' plane and facilitates creating horizontally-aligned details

Traditional modeling tools like Maya and SketchUp separate navigation from modeling, while many games allow players to interact with objects and shoot weapons while they move and look around. In most modeling tools, changing the viewing angle takes place in a different mode than creating or interacting with geometry, and the user must consciously switch between modes. When modeling is part of a first-person game such as Minecraft or the physics simulation sandbox Garry's Mod from Valve [10], navigation and object manipulation are the same mode. The player 'looks' at an object and clicks on it to interact with it. Due to the confusing visual nature of point clouds, combining interaction with first-person navigation allows depth cues from motion to help the user make sense of the point cloud. As users work, they can quickly check their modeling work from different angles as they go. If they make an error, they can seamlessly undo and try again without switching back and forth between navigation and modeling.

In the following section, we describe the first-person tools for building geometry and interacting with the point cloud.

3.2 Placing Pellets with the Pellet Gun

The most basic modeling operation in PointCraft is placing handles in the point cloud. We provide users with a shooting mechanism that lets them aim crosshairs in the center of the screen, and click to fire pellets at the nearest point in the point cloud, thereby specifying where to create these handles.

The *pellet gun* shoots spherical pellets that animate towards the point cloud, checking for intersections with nearby points and pellets, and stick to the point cloud at the first location they hit. When the user shoots, they see a spherical pellet (circle) emanating from them position in the direction they are currently looking. The pellet flies off toward the point cloud, eventually hitting and sticking to it. Since the pellet is represented by 3D geometry, perspective foreshortening causes it to shrink as it flies farther away, thereby conveying a sense of distance. The pellets can also stick to other pellets and user-created scaffolding. If a new pellet hits an existing pellet, the pellets combine into a single spherical pellet.

The pellet also provides auditory feedback upon impact, letting the user know that the pellet has landed. The delay before this sound provides a secondary indicator of distance. Additionally, different noises indicate whether the pellet has snapped to a point in the point cloud or an element of the user-created scaffolding.

3.3 Modeling Tools

PointCraft provides a number of *construction* and *editing* tools that are quickly accessible through the number keys 1–9 (a built-in online sharing tool is mapped to number key 0). Construction tools generate pellets and/or primitives, whereas editing tools operate on existing pellets in the scene. Every construction tool fires pellets of a different color (shown in Figure 5). Firing a new pellet at an existing pellet combines the two, and the new pellet snaps to the old one's position.

3.4 Polygon Tool

The most basic tool is the polygon tool, which shoots pellets to form vertices and connects them by edges. By firing a sequence of four pellets that starts and ends with the same location, the user can form a triangle. Including more pellets in this sequence results in the construction of a triangle fan around the first point, allowing the user to construct arbitrary planar polygons without switching tools.

3.5 Scaffolding Tools

Given a point cloud with no missing data, a user could construct a mesh using only the polygon tool. However, the resulting shape might be sloppy, like tracing a picture without a ruler. Thus we introduce scaffolding tools, which not only make the reconstruction neater, but also allow the user to extend the model past the edge of the point cloud. The simplest piece of scaffolding is a **single pellet**, which can be used to define polygons or other scaffolding. There are also **lines** (defined by two points) and **planes** (defined by three points). Once scaffolding has been defined, the user can fire pellets at a line or plane in addition to the point cloud. In Figure 4, the user has used line scaffolds to create even, regular dormer geometry and defined the shape of the roof despite the point cloud containing no roof data.

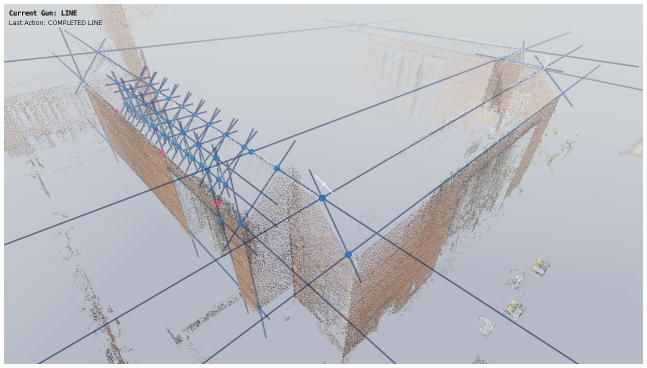
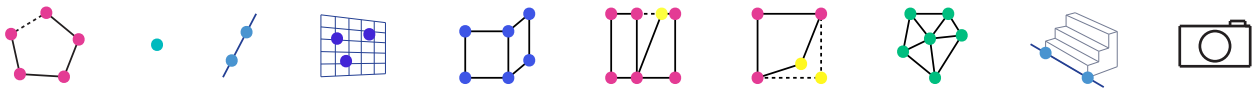


Figure 4: Line scaffolding for roof, walls, and repeated dormer structures.

3.6 Smart Tools

PointCraft's simplicity and versatility come at the price of speed. A user can create any mesh, but to do so using the above tools, they have to place and connect every pellet. In order to minimize tedious, low-level activities, we offer some smart high-level tools to accomplish complex tasks.



(a) #1 (b) #2 (c) #3 (d) #4 (e) #5 (f) #6 (g) #7 (h) #8 (i) #9 (j) #0

Figure 5: These are the nine PointCraft tools accessible through the number keys 1–9. From left to right, they are: polygon, pellet scaffold, line scaffold, plane scaffold, wall, combine pellets to edit, drag to edit, triangulated mesh, and change direction of wall tool. The 0 key is a tool that lets users select a viewpoint and automatically share a screenshot and their in-progress geometry.

In an effort to make reconstructing a building as easy as marking out its floorplan, we created the **wall tool**. The wall tool provides a shortcut whereby when the pellet hits the point cloud, it spawns two new pellets traveling in opposite vertical directions that stop when they find a gap in the point cloud, creating a line that spans the height of the point cloud at that location. The next time the wall tool is fired, it creates a new line attached to the pellet's impact location that is parallel to the first line and connected into a rectangle. Essentially, a user can create even wall segments with very few clicks. Figure 7 shows walls constructed with this method.

Since point clouds are not always perfectly vertically aligned, PointCraft allows the user to specify what 'vertical' means, in the context of the wall tool. By shooting a 'direction picker' pellet at any line, the user can choose that line's direction for the wall tool, thus turning the wall tool into a generic parallel line tool to quickly make stairs or slanted walls.

For shapes that are more curved and organic, we created a **meshing** tool, which computes the 2D triangulation of the 3D points. We used this to manually create coarse meshes of the bushes surrounding the building in Figure 9.

Once we were committed to the Minecraft-style first-person modeling paradigm, there were many more highly specialized tools we could imagine, such as tools specifically for stairs or windows, or tools that let users create regular circles, spheres, and boxes. Some of these regular-primitive tools we built and made available in an inventory inspired by Minecraft's tool and material inventory. From the tool inventory, users can choose which tools they want in their on-screen palette mapped to specific number keys.

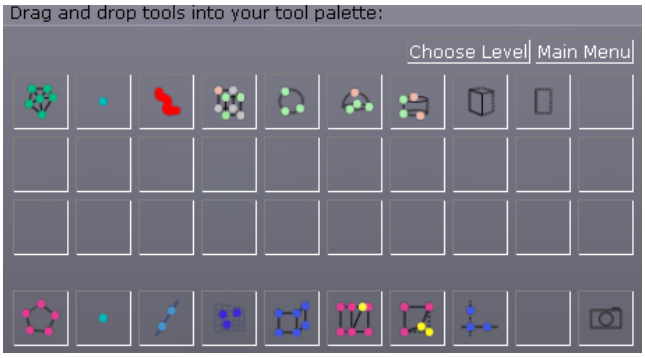


Figure 6: PointCraft users are not limited to nine tools; an inventory of additional tools lets them choose which tools they want to have immediately available through specific number keys. The additional tools include a paintbrush tool for selecting points, tools that make regular boxes, circles, domes, and cylinders, and tools that let the user extrude lines and polygons.

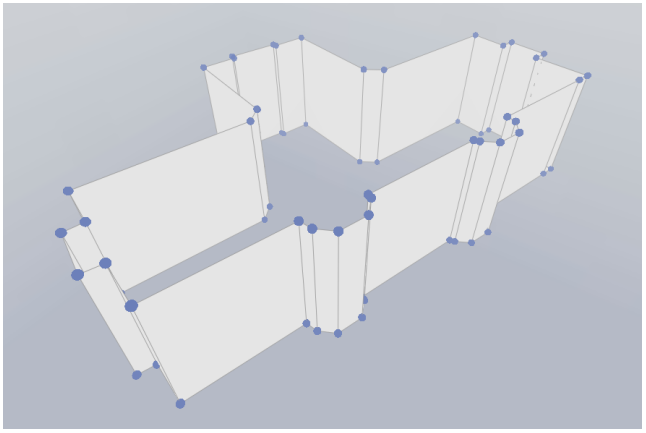


Figure 7: The wall tool makes constructing straight walls as easy as tracing a floorplan.

3.7 Editing Tools and Undo

We provide a small set of editing tools that are powerful, yet intuitive: dragging to edit the positions of the pellets and snapping them to other pellets.

We also provide unlimited undo via 'Ctrl-Z', which is extremely useful for quickly correcting errant pellets and polygons. The pellet gun may not position the pellet where the user intended, especially if there were unnoticed, closer point cloud points that the pellet stuck to instead of its intended target. But the integrated navigation and easy undo allows the user to quickly check the pellet's placement and easily try again from a different vantage point if the first attempt did not succeed.

3.8 Additional Controls

Users can toggle different display properties through several additional keys, such as hiding or showing the point cloud, pellets, or scaffolding, and changing the size of the points or the amount of fog. Adjusting the fog and point size helps when viewing the point cloud up close by providing additional depth cues. Because not all point clouds have the same size or point density, users can adjust the speed at which they move around the model, and the size of the pellets' collision radius.

3.9 Sharing and Collaboration

Built into PointCraft is the option to automatically share the model along with a screenshot. The user enters the camera tool mode, navigates to a nice viewpoint and clicks, whereby PointCraft uploads a file containing the user's geometry, along with a screenshot from that viewpoint. The screenshot is shown on the PointCraft website for others to download. Other users, or even the same user,

can load existing geometry along with the relevant point cloud and continue modeling where the original user left off. Figure 10 shows a model built by multiple people working on different parts of the same point cloud. In the current iteration, PointCraft does not support multiple simultaneous users, although it could, if we designed suitable avatars for players to be aware of each other's actions. We would likely borrow Minecraft's model in which players see each other as characters in the world, and can interfere with each other's work, but where social etiquette mostly keeps players collaborating. In the future, if there were a source of urban-scale point clouds that needed to be converted into clean, low-polygon geometry, we believe PointCraft users could collaboratively tackle this task.

4. RESULTS

Because PhotoCity was offline at the time PointCraft was developed, we used models from the PhotoCity Archives [25], including the buildings of Uris Library (at Cornell University), Lewis Hall, and Red Square (both at the University of Washington). These point clouds were automatically generated from a few thousand player-captured photographs and contain hundreds of thousands of points each. Automatic reconstruction methods completely failed on these point clouds, generating surfaces like the ones shown in Figure 2. It would take expert-level knowledge and effort to repair these meshes with traditional 3D modeling tools. With PointCraft, users are able to trace the point clouds directly and even fill in appropriate geometry where there was no point data. The resulting geometric models are clean and concise.

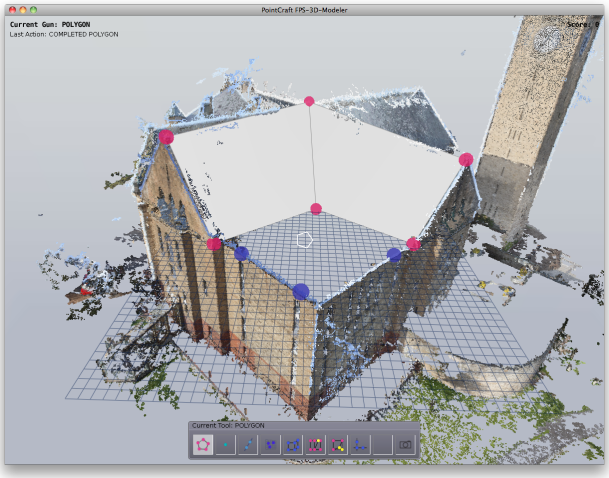


Figure 8: A plane scaffold being used to construct the roof where the point cloud contains no data. See Figure 1 for a view of the point cloud with no roof.

4.1 Lewis Hall and Uris Library (Cornell)

We now show author reconstructions of Lewis Hall and Uris Library, two models that were used as case studies when designing and developing PointCraft tools.

Because nearly all the images are taken from ground level, the point clouds do not include the roofs of the buildings. The model of Uris Library (Figure 1) demonstrates this lack of roof detail nicely. To model the roof in PointCraft, we constructed a planar scaffold and were able to place a pellet at a point on the roof that was invisible to all photos (see the pellet in the center of Figure 8). In total, it took an experienced user 65 minutes to construct the finished Uris model.

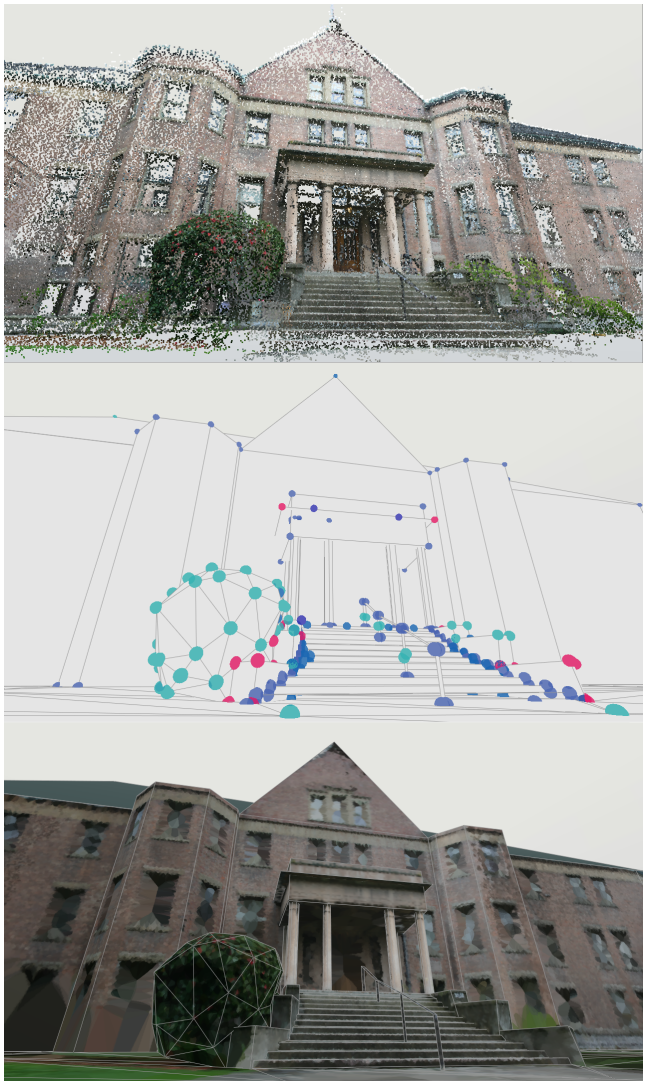


Figure 9: Lewis Hall model with pillars, stairs, handrail, and bush. The top image shows the point cloud. The middle image shows the PointCraft geometry and pellets. The bottom image shows the textured geometry.

The model of Lewis Hall had a simpler roof and simpler wall architecture, but had a lot of detail around the front entrance of the building that we wished to reconstruct. We modeled the walls (Figure 7) and roof of Lewis Hall in under 10 minutes, and spent another two hours on the entryway, including the pillars, stairs, stair railings, and foliage. Figure 9 shows the final result.

5. PILOT STUDY

In designing a modeling tool for tracing point clouds, we borrowed navigation and modeling paradigms from Minecraft and adapted them to point clouds. To make sure we had not created another 'experts-only' modeling tool, we ran a small pilot study and gave PointCraft to several novice users who had never used PointCraft before, had no experience with point clouds, and limited or zero

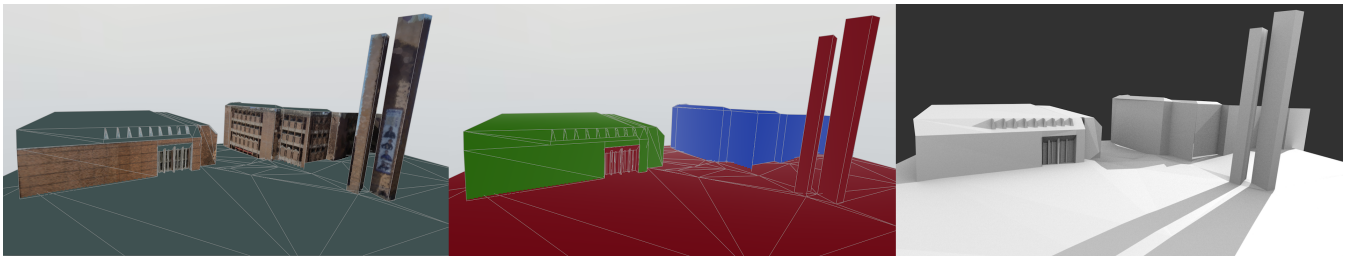


Figure 10: Multi-building model collaboratively built by multiple users.

experience with traditional 3D modeling tools. These users, four males in their twenties, did have experience with Minecraft and with gaming in general. We acknowledge that this is a small, biased sample and that there are many interesting questions remaining, such as skill transfer and how the game-inspired interface of PointCraft changes the overall experience of participating in such a game with a purpose.

During our pilot study, we sat down with four users and asked them to model a point cloud from PhotoCity of buildings in a location called Red Square on the University of Washington campus. We spent five minutes at the beginning going through each of the tools and letting each user scope out the scene. Following this, each user was allowed to choose what to work on, with no time limit given. They enjoyed the familiar FPS-style navigation and spent upwards of 45 minutes absorbed in their own modeling work. One of them, who had previously attempted to use Maya on his own but was unable to make any meaningful progress, praised the usability of PointCraft.

While each user worked individually, they modeled separate portions of the same multi-building point cloud. The collaboratively-built model combining their work is shown in Figure 10 with each user's work in a different color.

Even among these four users, we observed PointCraft?s small set of tools being used in different ways. One user set up scaffolding first and then spent little time editing (Figure 4 is the work of this user), while other users created freehand polygons and relied on editing for cleanup. This demonstrates an ability to accommodate more than one working style.

6. LIMITATIONS OF POINTCRAFT

PointCraft struggles with fine detail because it is designed for creating large, flat polygons. When a point cloud is a dense, clean approximation of a surface, automatic methods such as Poisson Reconstruction [12] work well, but many point clouds that people wish to use with Poisson are not complete. PointCraft's strength is laying out coarse geometry to fill in missing regions, so perhaps it can be used to aid the automatic Poisson method.

We used PointCraft to complete a model of the Sistine Chapel reconstructed from tourist photographs. Naturally, most people take photos of the ceiling, and the presence of other people obscures the view of the ground, so the ground is not included in the automatic reconstruction. Using PointCraft, we were able to model planes to extend the walls and complete the floor. Figure 11 shows the new points sampled from PointCraft geometry in white and Figure 12 shows a view from inside the Sistine Chapel model after running Poisson reconstruction with the floor now included.

Figure 11: PointCraft was used to add coarse geometry and sample new points along the walls and base of the incomplete Sistine Chapel point cloud.

Figure 12: The result of running automatic Poisson Surface Reconstruction on a point cloud of the Sistine Chapel augmented with points sampled from wall and floor geometry added using PointCraft.

7. CONCLUSION AND FUTURE WORK

Driven by the need to generate concise meshes from PhotoCity's point clouds, we have developed PointCraft, an interface for interactively tracing point clouds in 3D. Following our aim to retarget videogame players' existing skills, we have borrowed ideas from the first-person voxel world-building game Minecraft, designing a game-like UI that is familiar to such users. We have thus not only enabled existing players who are invested in PhotoCity to contribute, but also opened the door for new players to contribute to this part of the overall goal of reconstructing the world in 3D. PointCraft was successfully used to model PhotoCity point clouds, closing the gap between crowdsourced photographs and clean, concise 3D geometry.

By designing a game-inspired modeling interface with first-person navigation and shooting, we were able to harness players skills with previous videogames to a new, serious purpose. Furthermore, the first-person modeling paradigm seen in Minecraft worked exceptionally well with point clouds, which are otherwise difficult to navigate and interact with.

In the future, we hope to integrate PointCraft into the PhotoCity game. Our long term goal is to unify the two systems and see effort, ideas, and creativity flow back and forth between photographers and modelers. By creating more ways for players with different skills and interests to participate, games with a purpose will be able to attract a wide audience where many players with different interests and skills can collaborate on the same goals.

8. REFERENCES

[1] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *Proc. of the fourteenth annual symposium on Computational geometry*, SCG '98, pages 39–48, New York, NY, USA, 1998. ACM.

[2] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *Proc. of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266, New York, NY, USA, 2001. ACM.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, and S. Member. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5:349–359, 1999.

[4] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *CVPR*, pages 1261–1268. IEEE, 2010.

[5] A. Colburn, A. Agarwala, A. Hertzmann, B. Curless, and M. Cohen. Image-based remodeling. *Visualization and Computer Graphics, IEEE Transactions on*, 19(1):56–66, 2013.

[6] S. Cooper, F. Khatib, I. Makedon, H. Lu, J. Barbero, D. Baker, J. Fogarty, and Z. Popovic. Analysis of social gameplay macros in the foldit cookbook. In *FDG '11: Proceedings of the 5th International Conference on Foundations of Digital Games*, Bordeaux, France, 2011. ACM Press.

[7] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM.

[8] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. of the 23rd annual conf. on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 11–20, New York, NY, USA, 1996. ACM.

[9] C. B. Eiben, J. B. Siegel, J. B. Bale, S. Cooper, F. Khatib, B. W. Shen, F. Players, B. L. Stoddard, Z. Popovic, and D. Baker. Increased diels-alderase activity through backbone remodeling guided by foldit players. *Nature biotechnology*, 30(2):190–192, 2012.

[10] Facepunch Studios. Garry's mod. `http://garrysmod.com/`.

[11] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1422–1429. IEEE, 2009.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. of the 19th annual conf. on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 71–78, New York, NY, USA, 1992. ACM.

[13] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[15] J. Lee, W. Kladwang, M. Lee, D. Cantu, M. Azizyan, H. Kim, A. Limpaecher, S. Yoon, A. Treuille, and R. Das. Rna design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences*, 111(6):2122–2127, 2014.

[16] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: consistently fitting primitives by discovering global relations. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 52:1–52:12, New York, NY, USA, 2011. ACM.

[17] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Trans. Graph.*, 29:93:1–93:10, July 2010.

[18] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proc. of the 28th annual conf. on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 433–442, New York, NY, USA, 2001. ACM.

[19] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Technical section: Sketch-based modeling: A survey. *Comput. Graph.*, 33:85–103, February 2009.

[20] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 641–650, New York, NY, USA, 2003. ACM.

[21] M. Persson. Minecraft statistics. `https://minecraft.net/stats`. Accessed: 2015-04-20.

[22] R. Schmidt, A. Khan, K. Singh, and G. Kurtenbach. Analytic drawing of 3d scaffolds. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 149:1–149:10, New York, NY, USA, 2009. ACM.

[23] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[24] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 159:1–159:10, New York, NY, USA, 2008. ACM.

[25] K. Tuite. Photocity archive. **`http://photocitygame.com`**. Accessed: 2015-04-20.

[26] K. Tuite, N. Snavely, D.-y. Hsiao, N. Tabing, and Z. Popovic. Photocity: training experts at large-scale image acquisition through a competitive game. In *Proc. of the 2011 annual conf. on Human factors in computing systems*, CHI '11, pages 1383–1392, New York, NY, USA, 2011. ACM.

[27] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: rapid interactive scene modelling from video. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[28] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Request Permissions, Apr. 2004.

[29] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Proceedings of the First Eurographics conference on Point-Based Graphics*, SPBG'04, pages 85–94, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[30] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proc. of the 29th annual conf. on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 322–329, New York, NY, USA, 2002. ACM.