Penspective: a Perspective-Aware Game Map Editor with Natural Interaction

Alexandre M. F. de Sousa, Anderson C. M. Tavares, Carlos H. Morimoto
Department of Computer Science - DCC
Institute of Mathematics and Statistics - IME
University of Sao Paulo - USP

{*alemart, acmt, hitoshi*}@ime.usp.br

***Abstract*—Current WIMP (Windows, Icons, Menus, Pointers) interfaces used in most computer game map editors impose complex interactions to level designers. In this paper we present a novel map editing tool based on a Natural User Interface (NUI) that uses common drawing objects, such as pens and erasers, to build and edit map elements, as well as hand gestures to manipulate them. As a proof-of-concept, we have implemented a tile-based map editor featuring a 3D NUI with multitouch capabilities. The real-time prototype, built using two Microsoft Kinect devices and a regular LCD screen, displays the levels in 3D while adjusting the perspective according to the location of the user.**

***Keywords*—level design; sketching; natural interaction; 3D user interface; post-WIMP; off-axis perspective projection; depth-sensing; Kinect.**

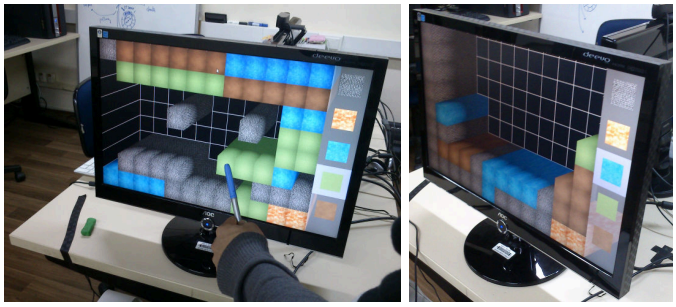


Fig. 1: Penspective.

I. Introduction

Map editors play an important role in the process of creating video-games. They provide a user interface that allows designers to build levels, defined as the virtual spaces in which the player interacts and plays. These programs can be distributed as stand-alone solutions, or can be bundled with game engines: software systems that provide a set of resources, including code abstractions that deal with low-level routines (graphics, sounds, asset management, etc.), to simplify the process of developing games [1].

Currently, most map editors use WIMP (Windows, Icons, Menus, Pointers) interfaces. During the process of designing levels, the user will often rely on cluttered menus or hotkeys to perform frequent operations such as inserting or transforming certain map elements. By being restricted to using devices like the mouse, keyboards, joypads or even touchscreens, current map editors impose complex interactions to level designers. The complexity of the user interfaces is also increased, as they demand that users perform many steps to accomplish certain goals.

In *Human-Computer Interaction* (HCI) research, advances in computer hardware and software stimulated the development of a new generation of user interfaces: these are the post-WIMP interfaces [2]. According to van Dam, they contain at least one interaction technique that isn't based on regular 2D widgets. The motivation behind these interfaces is to leverage the pre-existing, mundane knowledge of the users, in order to make the communication to the computers more similar to activities of the non-digital world. By employing elements of the users' everyday knowledge, we can design interfaces that reduce the gap between human goals and the actions required to accomplish them [3].

A common way of drawing sketches is to use pen and paper. These well-known objects are very versatile and widely used. Since they are present almost everywhere, using them feels like second nature. This idea can also be applied to map editing, reducing the complexity of the interfaces and enabling more natural interactions.

Besides interaction, visualization also plays an important role for usability and user experience in games [4]. 3D games can increase the user degree of immersion. However, the immersion may be reduced if the system does not use information from the real environment within which the user stands. The sensation of depth in a 3D scenario improves if the environment adjusts its perspective according to the viewpoint of the user.

In this paper, we introduce Penspective, a novel tile-based map editor that builds upon the idea of using a pen to draw level elements on a grid paper, an eraser to remove them, and finger movements to transform them. In our proof-of-concept, the levels are built on a grid, where each cell holds a 3D block with adjustable height. Our prototype, shown in Figure 1, is displayed on a regular LCD screen, transforming it into a 3D natural user interface with multitouch capabilities. The system displays the maps in 3D while adjusting the perspective according to the user's head, giving the illusion that the map is actually lying behind the screen. The motivation of this work is to present a new form of interaction for map editing, providing level designers a "natural" way to perform their tasks.

II. Related work

Map editors have been around for decades. Older editors used to create tile-based maps for 2D games include Games Factory [5] and Mappy [6]. Newer alternatives include Tile Studio [7] and Tiled [8] (both are free software). Although they contain varying features, they all draw upon the same interaction: one will use a mouse to select certain tiles and place them on the level.

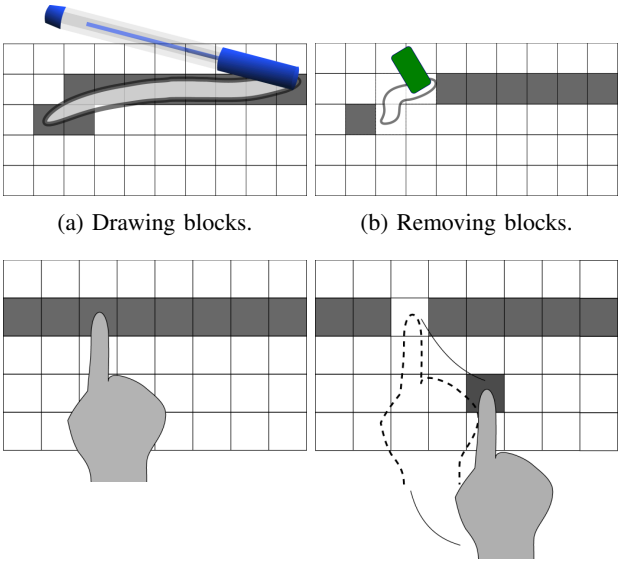
Designed for building 3D virtual worlds, other alternatives include: Ambierra's irrEdit [9] and Unity3D [10]. The latter is a popular cross-platform game engine that includes a built-in 3D world editor, while the first is an editor distributed as a standalone application. In spite of the advantages of these solutions, both make heavy use of the mouse, demanding that users spend considerable time manipulating the interface as they perform sequences of operations (e.g., selecting appropriate tools in a menu, manipulating certain widgets to fine-tune parameters of the map, etc.) to achieve their design goals. In addition, the mapping between 3D tasks and 2D controls is less natural compared to what one would experience in a purely bidimensional application, thus increasing indirection and cognitive distance [2].

In the field of computer-aided architectural design, there is work exploring non-traditional user interfaces. In an attempt to intuitively convey the designers' goals, Mistry *et al.* developed *Inktuitive* [11], a tool that lets them create 3D objects while taking advantage of the intuitive process of using pen and paper. As the user draws an outline of the desired object on a paper, the drawing is captured by a computer and transformed into a polygonal entity in the digital world. The user can then lift the pen above the paper to specify the height of the object, which is viewed on a vertical LCD screen. Lee and Ishii's *Beyond* [12] allows users to combine collapsible physical tools with hand gestures in "beyond the screen" 3D design. The designers will hold either a pen to draw forms, or a saw to perform sculpting and cutting. Hand gestures are used, as modes of operation, to complement the direct manipulation performed with the tools.

III. Design

Under the WIMP paradigm, it's not unusual that relatively complex applications such as map editors include a lot of widgets. Complex interfaces that have too many widgets cluttering up the screen may turn the application hard to learn and hard to use. With that in mind, we have taken design decisions that leverage advantages of physical tools and users' pre-existing knowledge, attempting to explore natural ways of designing maps for games.

Our map editor is inspired by the notion of grid papers. Similarly to any regular paper, one is capable of using a pen to draw elements on the paper, as well as removing them with an eraser. Due to the fact that our system supports physical artifacts representing different tools, there is no need of an additional menu to select the desired tool. As the indirection decreases, so does the complexity of the interface. Rather than adding more modes of operation, the use of an actual pen and an actual eraser provide obvious drawing metaphors, revealing a seamless bridge between the physical and the digital realms.



(a) Drawing blocks. (b) Removing blocks.

(c) Moving (selection phase). (d) Moving (dragging phase).

(e) Adjusting height (before). (f) Adjusting height (after).

Fig. 2: Interacting with the map editor.

Initially, the editor displays an empty grid. Each cell of the grid can be filled with a block corresponding to materials such as: wall, fire, grass, water or dirt. These blocks can have their heights adjusted by the designer, so that 3D maps can be built. As depicted in Figure 1, the system also provides an illusion of depth. Like a magic paper, it's as if the 3D map is actually present in the physical world.

Our map editor supports the following operations:

1) **Drawing:** similarly to a drawing activity, as the designer drags the pen over the surface, blocks are placed on the map. Figure 2a illustrates the process;
2) **Erasing:** erasing works in the same fashion as drawing. As the user touches the surface with the eraser, the corresponding blocks are removed from the map. Figure 2b shows the operation;
3) **Moving:** human beings frequently use their hands to grab and push physical objects. In our editor, blocks are moved using a finger (see Figures 2c and 2d). Touching a block with a finger starts the operation, selecting that element. Then, the selected block is moved around the map as the user drags the finger over the surface. The block is placed on its final position when the user no longer touches the surface;
4) **Selecting a block type:** on the right of the screen

there is a panel featuring a few block types. The designer selects one of them to change which block types are placed on the map during the drawing activity. This operation is akin to the notion of changing the ink of the pen;

5) **Adjusting height:** the height adjustment is done right after new blocks are drawn over the map. The designer starts the operation by keeping the pen in contact with the surface and placing a finger on the screen. Then, lifting the pen changes the height of the recently drawn blocks, as depicted in Figures 2e and 2f. The operation finishes once the finger is lifted. With that gesture, designers can use their dominant hand to adjust the height of the blocks, as pens are frequently holded with that hand.

IV. System description

The system is composed by four components: Multitouch Interaction, Above-The-Surface Interaction, Skeleton Tracking and Main Application. The Multitouch Interaction module turns a flat surface into a touch sensitive region. That module is closely related to the Above-The-Surface Interaction subsystem, responsible for tracking, in 3D space, physical objects that were previously in contact with the surface. The Skeleton Tracking subsystem tracks the user in physical space. Finally, Main Application uses the other subsystems to actually present the map editor to the user. The following sections describe the subsystems:

A. Multitouch interaction

The advent of low-cost RGB-D cameras has enabled the detection of touch events on ordinary surfaces. Using a Microsoft Kinect, Wilson [13] has described an image processing technique that detects touch on a non-instrumented tabletop. Although the results are not as accurate as they would with capacitive touch screen technologies, the approach is good enough for various applications, with the added benefit that the physical space above the surface may also be exploited.

In this project, a Microsoft Kinect sensor is used to turn a regular LCD screen into a multitouch surface. Additionally, the surface is also capable of detecting which objects touch it (be it a pen, an eraser or a finger). Given that the Kinect also includes a color camera, by extracting the colors of the neighborhood of the touch points, one is capable of recognizing the corresponding physical artifacts using a variation of Lee and Yoo's elliptical boundary model [14].

B. Above-the-surface interaction

By using a Kinect device instead of a regular touchscreen, we devised a technique so that the user is also able to interact in the 3D space above the surface. Depth and color cameras are again combined. Initially, the color image is converted to grayscale. Once the user touches the surface with his/her finger or with other physical artifact, the grayscale image is used in order to take a rectangular template around the area in which the touch event had ocorred. The background is removed using depth data. Then, the touching object is tracked using a weighted template matching technique.

Given a fixed lookup radius r and an empirically determined factor $\alpha \in [0, 1]$, the pair p_t of coordinates corresponding to the top-left position of the template in time t will be given by picking (x, y) that minimizes the function:

$$cost_t(x,y) = \begin{cases} \alpha \cdot (1 - R(x,y)) + & if\ d_t(x,y)\ \leq\ 1 \\ (1-\alpha) \cdot d_t(x,y) & \\ & \\ \infty & otherwise \end{cases}$$

where $d_t(x, y)$ is the normalized Euclidean distance $\|(x, y) - p_{t-1}\| / r$ and $R(x, y) \in [0, 1]$ is a normalized matching value computed through fast template matching routines, using correlation coefficient methods [15]. $R(x, y)$ increases as the matching between the stored template and the data at position (x, y) of the current image improves. As the touching object is tracked in the grayscale image derived from the color camera, its distance is extracted through the depth image.

C. Skeleton Tracking and Off-Axis Perspective

This module tracks the user head for the *Off-Axis Perspective* Projection. The first step is the calibration. The calibration is done once by tracking 4 points: the right shoulder and the right arm directed to the right, up and front. These points are input for the Singular Value Decomposition (SVD) method to get the transformation matrix which maps coordinates from the device to the world coordinates. Ho [16] explains how to get the rigid body transformation matrix, decomposed into translation and rotation matrices. The origin of the world coordinate system coincides with the point representing the bottom-left corner of the screen.

Off-Axis Perspective projection is responsible for creating an illusion of depth for the scenario, adjusting its projection by using the position of the user head. Kooima [17] explains the process. The portion of the scene projected onto the screen does not depend on the orientation of optical axis of the user ocular system [18].

The technique needs 3 points from the surface (the bottom-left, bottom-right and top-left corners) to calculate the 3 axis representing the screen orientation. These points, along with the user eye position, form an assymmetrical pyramid. The perpendicular projection of the eye to the surface is used to calculate the parameters needed for the projection matrix. The pyramidal volume of the projection is assymmetrical, thus incurring in an oblique projection, whose necessary steps are described by Majumder [18].

V. Results

Figure 3 shows a prototype of Penspective. The user is able to draw maps within a grid of fixed size. In order to give the illusion of depth, the displayed graphics change according to the viewpoint of the user. We use a blue pen and a green eraser as the physical tools. Blocks can be drawn with the pen and removed with the eraser and moved with a bare finger. The height adjustment operation can also be performed.

The hardware setup features two Microsoft Kinect devices and a 23" LCD monitor. Our prototype, developed in C++,

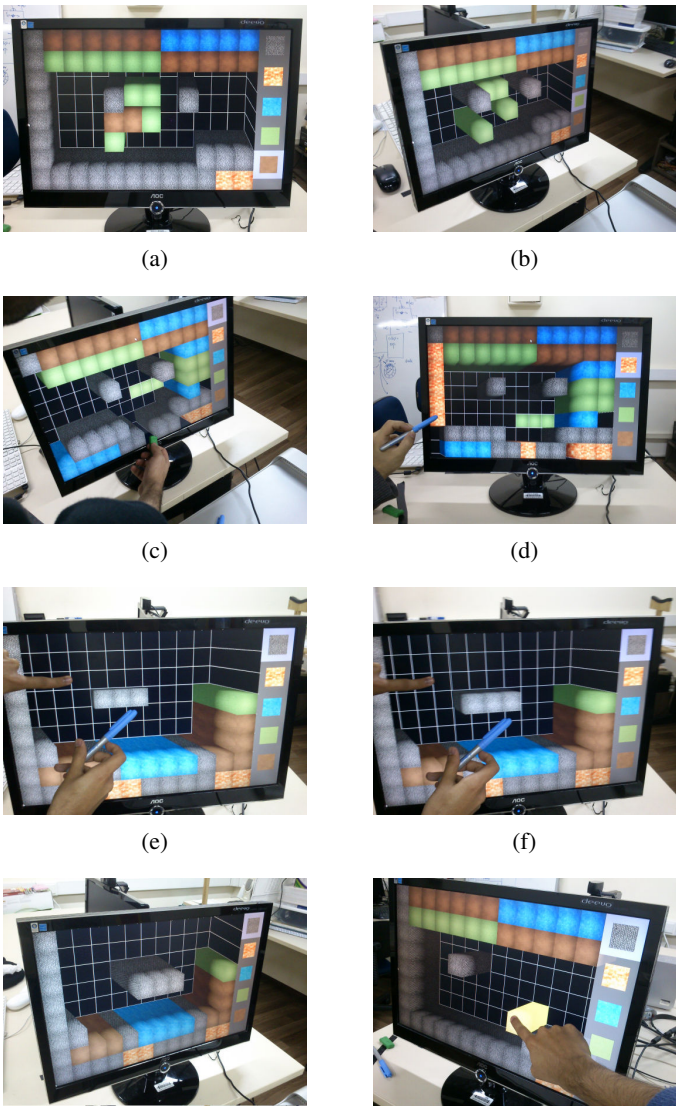


Fig. 3: Prototype (a,b) changing the viewpoint, (c) erasing, (d) drawing, (e,f,g) height adjustment, (h) moving a block.

has been tested using an Dell Vostro 4350 laptop (featuring an Intel Core i5 2.50 GHz dual-core CPU and running Ubuntu GNU/Linux 12.10) and an iMac (Intel Core 2 Duo, 2.66 GHz CPU running Ubuntu 13.04), both connected to a wired local network. In our setup, the iMac performs the skeleton tracking, while the laptop is responsible for the other tasks. One of the Kinect sensors has been placed at about 80cm far from the LCD screen, pointing to it. The other Kinect tracks the user and has been placed at about two meters from the person. The subsystems presented in Section IV exchange data using *Virtual Reality Peripheral Network* [19].

VI. Conclusion

Envisioning more direct ways of building maps compared to WIMP-based alternatives, we have presented a form of interaction that combines simple gestures with the intuitive idea of drawing on grid paper. Using physical tools such as pen and eraser, we have designed a 3D natural user interface to create maps for games. We have also built a working prototype in order to show the viability of the concept. Preliminary tests show that the user is capable of moving, drawing, erasing and adjusting the height of 3D blocks in a map of fixed size. The rendering is adjusted according to the location of the user, giving an illusion of depth. Future work includes the design of more advanced gestures for crafting maps and the conducting of a more in-depth user study.

References

[1] A. S. Perucia, A. C. de Berthêm, G. L. Bertschinger, and R. R. C. Menezes, "Desenvolvimento de jogos eletrônicos," *São Paulo: Novatec*, 2005.

[2] A. van Dam, "Post-WIMP user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–67, Feb. 1997.

[3] R. J. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum, "Reality-based interaction: : a framework for post-WIMP interfaces," in *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, ACM. New York, New York, USA: ACM Press, 2008, p. 201.

[4] B. Bowman, N. Elmqvist, and T. J. Jankun-Kelly, "Toward Visualization for Games: Theory, Design Space, and Patterns," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 11, pp. 1956–1968, Nov. 2012.

[5] E. S. Martinez, "Games Factory," 2013. [Online]. Available: http://www.arrakis.es/~esanchez/

[6] TileMap, "Mappy." [Online]. Available: http://www.tilemap.co.uk/mappy.php

[7] M. Wiering, "TileStudio," 2012. [Online]. Available: http://tilestudio.sourceforge.net/

[8] T. r. Lindeijer, "Tiled," 2013. [Online]. Available: http://www.mapeditor.org/

[9] Ambierra, "Ambierra irrEdit," 2013. [Online]. Available: http://www.ambiera.com/irredit/

[10] U. Technologies, "Unity3D," 2013. [Online]. Available: http://www.unity.com

[11] P. Mistry, K. Sekiya, and A. Bradshaw, "Inktuitive: an intuitive physical design workspace," in *4th International Conference on Intelligent Environments (IE 08)*. IET, 2008, pp. P11–P11.

[12] J. Lee and H. Ishii, "Beyond: collapsible tools and gestures for computational design," in *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*, ACM. New York, New York, USA: ACM Press, 2010, pp. 3931–3936.

[13] A. D. Wilson, "Using a depth camera as a touch sensor," in *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*, ACM. New York, New York, USA: ACM Press, 2010, p. 69.

[14] J.-Y. Lee and S. I. Yoo, "An elliptical boundary model for skin color detection," in *Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology*. Citeseer, 2002.

[15] A. Kaehler and G. Bradski, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O'Reilly Media, 2008.

[16] N. Ho, "Finding Optimal Rotation and Translation Between Corresponding 3D Points," 2013. [Online]. Available: http://nghiaho.com/?page_id=671

[17] R. Kooima, "Generalized Perspective Projection," 2013. [Online]. Available: http://csc.lsu.edu/~kooima/articles/genperspective/index.html

[18] A. Majumder, "View-Perspective Projection," 2011. [Online]. Available: http://www.ics.uci.edu/~majumder/CG/classes/wk3-cls1-persp.pdf

[19] R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser, "VRPN: a device-independent, network-transparent VR peripheral system," in *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '01*. New York, New York, USA: ACM Press, 2001, p. 55.