Authors: Grit Schuster, Bert Freudenberg (and everybody else at Impara)
Affiliation: impara GmbH, Listemannstr. 10,
39104 Magdeburg/ Germany

URL: http://planet-plopp.de (a demo version can be provided on demand)
Keywords: 3D, children, product development
Dialect: Squeak 3.8

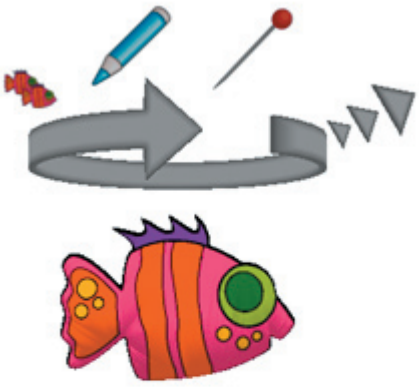
Plopp – A Squeak Application on the Shelf

Plopp is a 3D painting tool for children age six to ten. Plopp allows children to create 3D worlds without using complex 3D modelling techniques. Regular 2D paintings are inflated (thus the name Plopp). The 3D objects can then be moved, rotated, or scaled. Children can paint their own floor and background and also adjust the lighting. Other features of Plopp are the ability to send the image of the scene as an ecard, to print it or use the scene as the screen background. An easteregg feature is the ability to create an animation out of a sequence of world pictures by exporting them as an animated GIF.

User Interface challenges - Turning 2D into 3D

Plopp's core algorithm is based on Takeo Igarashi et al.'s work in the Teddy interface, a sketch-based 3D modeling software [2]. We modified the current Croquet version of the algorithm to become more robust. Typical problems occurring with naive users are small transparent holes or single opaque colour pixels in it. We use opening and closing algorithms known from the area of image processing that consist of erosion and dilatation working on the pixel level. Small structures like single pixels or small groups of pixels are removed and small transparent holes are closed by setting the pixels' color to that of the adjacent opaque pixels.

Interacting in 3D



Another challenge was interaction in the 3D world. How can children manipulate 3D objects? We built a threedimensional halo-like object menu that allows to translate, rotate, scale, copy, edit, or delete the objects. As the menu always appears around the object, it was necessary to scale and translate it dependent on the size and position of the object. In the back of the scene the object menu needs to be scaled less than the object as it would become to small to be usable. At the front it might become too large or partially outside the viewport.

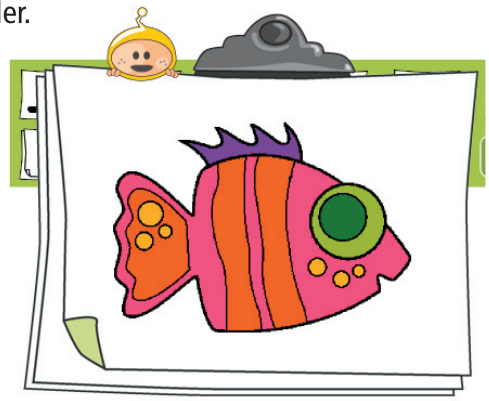
Picking a color

When designing a painting tool for children even the seemingly simple feature of picking a color requires a different solution. We decided to simulate subtractive color composition as most children already know how to do that with e.g. water colors. The user interface consists of five color tubes: the three primary colors red, blue, and yellow, plus black and white. You can mix most of the colors using these five colors by sliding the ends of the tubes forwards or backwards.

Another enhancement is the ability to have two different textures for the front and the back of the painting. The sizes and especially the contours of the two textures have to be the same. Our solution is a painting tool with a „turn around" canvas so you can just see the „backside" of the painting. The two sides are treated as layers, every stroke you paint is a new layer. If you paint on pixels that were transparent before the pixels are also visible on the opposite side, making the contours of the two textures fit automatically.

Minimizing the use of written text

We designed a graphical user interface that almost entirely works without text and „bubble help" is spoken by a character named Plipp. Implementing audio bubble help actually turned out to be an unexpectedly complex problem and we ended up implementing a hierarchical, priority based audio controller.



Automated storage

Hiding the complexity of dealing with files, saving and organizing the artwork was a major concern in designing the application for children. Inspired by the database-like approach the Exobox team took for their applications we are using a versioned data storage. Plopp automatically saves

objects, backgrounds, and scenes, and groups them by versions. Children don't have to worry about losing anything because they forgot to save and they also can always go back to older versions of their work. In the user interface the objects, backgrounds and scenes are represented by thumbnails in drawers, ordered by modification date. Older files are stored in a junk box where you can rummage around for your older work.

System Integration Direct3D rendering for Croquet: A typical children's PC is already a few years old. We got us a small test park from eBay where it became obvious that OpenGL support on such older machines is very unreliable. Thus, we added a Direct3D renderer based on Balloon3D.

Application Directory Layout: We came up with a portable directory structure which keeps Mac, Windows, and Linux VMs, image and content data in a single place. On the Mac, you simply drag-and-drop the Plopp application icon from the CD to your application folder, and double-click it to run. The very same icon appears as directory on Windows and Linux. Once copied to the hard-drive, double-clicking Croquet.exe or Croquet.sh in that directory launches Plopp.

File Locations Typically, Squeak applications store their data in the image directory. However, there are designated locations in most OSes for images, preferences, and other user data. We use FFI calls to determine those locations and store our files there. This also enables Plopp to be run directly from the CD-ROM.

Anti-aliased wallpaper We render the scene in higher resolution (4x4 tiles) and down-sample to the desktop resolution. This gives a nicely anti-aliased image that is set as desktop wallpaper using FFI calls. On Linux, a user-customizable shell script takes into account the currently active window manager (KDE or Gnome).

Production Process On top of the technological and user interface problems, the building of a CD-ROM product in Squeak also held some challenges. We needed to integrate designers and coders into a short turnaround workflow. Monticello provided a valuable infrastructure for the coders. The daily, or sometimes hourly updates were made painless with the use of Monticello Configurations. We used Subversion for the artwork, with the coders checking out the newest versions directly into their build environment. Our skinning framework allowed us to start with stand-in art and later easily update the artwork without interfering with the coding process.

We created the tutorials by event recording a demonstration. The replay then rendered each frame into a PNG which then was converted to MPEG. The intro and outro animations were created in Flash and also converted to MPEG. The playback works nicely using Squeak's MPEGPlayer, which also avoids the hassle of distributing a Flash or quicktime player.

Sources

[1] http://planet-plopp.de/ [2] Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. ACM SIGGRAPH ,99, Los Angeles, 1999, pp. 409-416 http://www-ui.is.s.u-tokyo.ac.jp/~takeo/teddy/teddy.htm [3] http://opencroquet.org/