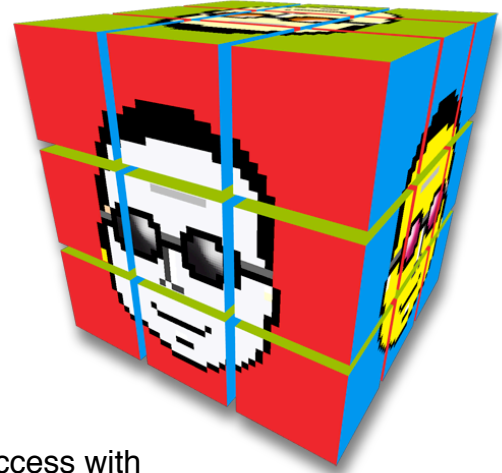


AgentCubes β

Boulder, Colorado Oct 12, 2011

Dear new AgentCubes users:



I would like to share my excitement about the release of AgentCubes beta. Over the years we have had great success with AgentSheets, becoming one of the first game design and simulation authoring tools so accessible that it has become *part of the computing curriculum* in many schools around the US and the world. Recently, the measurable impact on girls and underrepresented communities resulted in the presentation of AgentSheets as a success story reported to the director of the National Science Foundation. Of course, we will keep working on AgentSheets, but with funding from the National Science Foundation we took on an even more daunting challenge: how can we make 3D computing accessible to an extremely broad range of users?

Incremental 3D: 3D tools generally introduce a steep learning curve. How is this possible? As human beings living in a three dimensional world we should be well equipped to comprehend and manipulate 3D objects. 3D has become ubiquitous even in our electronically enhanced lives. Most games on consoles are 3D. We watch 3D movies in cinemas, and increasingly even at home. There is now even early evidence that 3D presentations have educational value allowing students to comprehend sophisticated concepts more easily. Yet, in all of these examples we are just *consuming 3D*.

At AgentSheets, as usual, we asked “How can we *create 3D*?” High end 3D tools have been around for some time. In the hand of trained experts these tools have been employed to create the wonderful 3D content we find in animations and movies. Unfortunately, few tools are made for end users with little time or interest to learn complex 3D authoring tools.

The fundamental challenge in 3D fluency is that most existing authoring tools conceptualize 3D as something that *begins and ends with 3D* manipulations. That is, from the first moment of using these tools users have to conceptualize everything in 3D. Where do I need to put the camera to actually see something? How do I make even a simple shape without getting lost in difficult interfaces including deeply nested menu hierarchies? If I want to make a game do I have to resort to having to choose from already existing 3D objects created by someone else or can I make my own 3D shapes? Imagine a completely different world in which tools would allow users to *start with 2D and incrementally move towards 3D*. We call this Incremental 3D. This is what AgentCubes is all about. You start with a simple 2D drawing tool and gradually turn these 2D images into 3D shapes. Create simple shapes such as a person, a bug, a fire,

a rock, a piece of chocolate or ...? Only your own imagination is the limit. Most likely, these 3D shapes may not win prizes but they are *your creations*. Next, you put these shapes (we call them Inflatable Icons) into a world. Place them into a grid, stack them up, create layers, switch to first person view, enter your own world... You can keep that world flat 2D, or turn that 2D world incrementally into a 3D world. In the end then, how much more complex is 3D compared to 2D? Perhaps not at all. Even with a very early version of AgentCubes our evaluation found that there were many cases where debugging in 3D was actually easier. For instance, imagine accidentally stacking up a very large number of agents. In AgentSheets, or most 2D tools for that matter, that problem may be hard to “see”. A thousand flat objects piled on top of each other still look like a single object. In 3D the problem is immediately visible. Perhaps, if we create the right kinds of incremental tools, 3D can ultimately be simpler than 2D!

Fundamental Authoring Challenges: When we set out to design AgentCubes we wanted to address what we found to be some fundamental 3D authoring challenges relevant to game design as well as simulation authoring:

- **Modeling:** How can we make 3D modeling substantially more accessible? The incremental 3D modeling allows users to gradually create 3D models from 2D images. Draw a frog with the symmetry tool in 2D. Now inflate it. Inflate part of your frog more selectively. Add noise for 3D texture. Make things transparent. Voila, just a minute later you have a 3D frog.
- **Animation:** It is not that hard to animate a small number of objects, but what if you have hundreds or even thousands? How are we to deal with dependencies? Imaging two objects heading to the same place. How will they pile up? What is the order in which the objects are piling up? Instead of having to worry about this an AgentCubes user just turns animation on and adjusts the time it takes with a slider. Problem solved, a thousand objects are animated.
- **Visualization:** Over time AgentSheets users have created sophisticated games and simulations. Some of the concepts used to build sophisticated game AI, for instance Collaborative Diffusion for collaborative agents, appear to be out of the reach of casual end user programmers. But, what if we make it really simple to visualize phenomena such as diffusion in real time and integrate these visualizations into the game world? The results are not only intriguing scientific visualizations but the powerful illustration of causality that can make advanced concept such as diffusion accessible even to elementary school students.
- **Programming:** We created one of the *first* educational drag and drop visual programming languages in 1994. But drag and drop just eliminates syntax errors and is not enough. How can a programming environment, especially one for 3D, help you with the meaning of your program? *Conversational Programming* executes your program one step into the future and shows you what your program would be doing. By visualizing the difference between the program you want and the one you have you can catch bugs even before your program is completed. Future Trace, the Conversational Programming 2.0 version built into AgentCubes even animates the consequence of changes in your game world resulting in the execution of different conditions/actions/rules and methods.

There are many features to AgentCubes but this letter is not the place to share them. I sincerely hope you will enjoy AgentCubes and help to shape it. AgentCubes beta is largely feature complete. I want to invite you not only to play around with AgentCubes beta but to share your insights and creations with us. Please use our forum, share your projects, even if they are incomplete, with use and others in the Scalable Game Design Arcade. Send me emails. Suggest things even if they sound crazy. Send me bugs (I hope there are not too many). Challenge any preconceived perception of what can be done with AgentCubes. Report limitations as well. The idea, of course, is to start thinking about where to go. The notion of Incremental 3D is research funded by the National Science Foundation exploring what can happen if 3D authoring is being made accessible to radically broaden participation. At the same time the National Science Foundation has given us a mandate to make AgentCubes a self sustaining project. We will have to charge for the tool to pay programmers to maintain, and further develop, AgentCubes. I hope you are OK with that. If you have suggestions for different business models please share them with us.

Beta Testing: If you find bugs please report them. We include a bug reporter that makes bug reporting as simple as possible. For instance, the bug report will include information about your OpenGL setting that will make it simpler to understand the circumstances of your problem. Feel free to share your project via the Arcade or send it via email to us if needed. We will release a number of beta updates and inform you of new versions. When you paid, you really paid, with a significant discount, for the final version, not the beta. We promise to work hard to make the final the tool you want to have. A list of known issues is below. There are more issues at the moment for Windows than for Macs. This is largely due to the fact that integration of OpenGL, which is the 3D library we use, is not as well supported on Windows as it is on the Mac. On Windows machines there is a good chance of older drivers, older versions of OpenGL libraries or tricky combinations of video hardware and driver/API software. We have tested on a number of machines but the real test will be to see how AgentCubes and OpenGL behaves on your machine. Please give us feedback if things do not work but also let us know when they do.

Sincerely,

Dr. Alexander Repenning