The Use of Mathematics in Video Games

Katharine Anderson

Coleman University

Abstract

The video game industry relies heavily on math, but not always in the way most people think. While math is a fundamental requirement for creating video games (which will be discussed later), math is also needed for supporting them. This includes customer service, quality assurance, and the tools needed for both. In this paper, the main focus will be on customer service and the creation of video games. It is assumed that the reader has some experience with both video games in general and massively multiplayer online (MMO) video games (i.e. EverQuest, World of Warcraft, PlanetSide 2, League of Legends, etc.). There were several interviews conducted in the research of this paper, and due to the competitiveness of the industry (and the fact that the author wishes to retain her current employment) the names of those interviewed and the company they are associated with will be altered. The original interviews will be included with the final assignment.

First, before any customer service can get involved, there must be tools. These include anything from logs detailing every single action done by players in-game, to look-up tools that can be used to find information regarding players accounts and/or characters, to basic tools that can allow customer service agents (generally known as Game Masters or GMs) and quality assurance testers (known as QA testers) to do things like rename a character or force a specific action to happen in-game. These tools are the backbone of video games. To create these tools, a large amount of math is required, but more in a logic sense than the use of complex mathematics.

M is a software engineer, and one of his main tasks is to create the tools used by the customer service team for a large online game. M says, "The concept of formal logic has been very important to me for programming, since it can help plan out and visualize what's happening in a computer program. My logic course was by far the most important because it showed me that the building blocks of computer programming were universal, logical concepts. Getting that knowledge was important because it let me understand why programming constructs are used the way they are and how to visualize difficult problems." This type of logic includes things like checking to see if a particular condition is met, and then running a specific section of code if it is.

As previously mentioned, one tool that is used by in-game customer support is a rename tool. The basic logic for a tool with this function may look like this:

```
bool isTaken = false;
for(int q = 0; q < database.charNames.total; q++)
{
	if(newCharName == database.charNames[q])
	{
		cout << "Name taken, please try a different name" << endl;
		isTaken = true;
		break;
	}
	else if(newCharName != database.charNames[q])
	{
		isTaken = false;
		continue;
	}
}
if(!isTaken)
{
	database.player.charName = newCharName;
	cout << "Name for " + database.player + " has changed to " +
		newCharName << endl;
}
```

This snippet of code runs through some basic logic of a rename. First it iterates through all the current names in a database. Then it looks to see if the current name is the same as the name the player wants. If the names is taken, it displays an error, sets a Boolean to true so that the program knows the name is taken, and then breaks out of the loop. If the name is not taken, it sets the same Boolean to false so the program knows the name is not taken, and then continues on with the loop. After the loop is complete, if the name is not taken, it sets the character name to the new name and announces it.

While this seems like very basic logic and algebra, but it is extremely important. The loop iterates through each name, and this is done with basic algebra. Each element in the array must be checked, the above uses an integer for both counting and the element number, which is covered by using the same variable used in counting through the loops.

Moving into the customer service side, the most obvious math usage is mainly done by supervisors and managers. S is the Director of Customer Service at a video game company, and she uses math, "when determining staffing (heads and timing), budgets, and contact deflection

needs." Essentially this means that based on contacts from players she must determine the budget necessary, and then the amount of staff needed to handle the calculated volume.

One great example of this is when a new game launches. Not only are the development, marketing and quality assurance teams busy, customer service must predict the possible issues and be prepared to handle them. In the past, this has been handled in mainly two ways: either a large number of people are hired for customer service (leading to being severely overstaffed and probably layoffs) or very few new people are hired (leading to being understaffed for several months but without layoffs). Each launch brings more information that is factored into how the next launch will be handled. Essentially, the possible influx of contacts from players (generally 200 – 500 contacts daily for a launch, spanning over several weeks) must be factored with the total number of GMs dedicated to this single product (anywhere from 4 – 40) and the time necessary to handle a single contact (normally 10 – 60 minutes). If there are 500 contacts a day with 10 GMs available and each contact takes 30 minutes, approximately 300 contacts can be handled over a single 24 hour period, not taking days off into account. This means that the number of contacts waiting will keep increasing over several weeks until either the issues causing contacts are resolved, or until players simply stop contacting support. With just 5 more GMs, the average number of contacts being handled daily goes up to 450, which is much better. However, when issues are resolved and fewer tickets are coming in, there would be too many GMs and thus possible layoffs. Finding the ideal number of GMs to have is extremely important, for both financial and basic customer service reasons.

Now for everyone’s favorite part, the actual development of video games. While it may seem obvious, math is extremely important when creating video games. Math is used for many different aspects of a game, including physics (gravity, movement, etc.), basic statistics (health,

score, kill/death ratio, etc.), and spatial awareness (coordinates, distances, 3D graphics, etc.). Physics are absolutely necessary, as without gravity or movement, the game would have no direction and make no sense. Basic character statistics, such as health and score, show a player's progression. Spatial awareness may be one of the most important parts of a game, especially with the popularity of 3D games.

3D graphics rely on geometry, trigonometry, and linear algebra. All 3D graphics are made up of multiple vertices, generally in a vector (a point in 3D space with direction and magnitude) which has three to four values on the coordinate plane, to create triangles (also called polygons or faces) that are drawn together to create an object. A cube would have 8 vertices to create 12 faces. A more complex object, such as a person or even a sphere, would have many, many more vertices, faces, and polygons. Not only is the shape itself complex, but so is moving it. When an object is moved (or translated) in 3D space, three axes (x, y, and z) must be considered in the equation. J is a generalist programmer on a fairly new title, and he says, "If I am handling a task that deals with the 3D environment, I would be lost at sea without the knowledge I learned from my linear algebra class."

When an object in 3D space is being handled, the type of transformation is extremely important. 3D objects can be translated (moved along the x, y, or z axis), scaled, and rotated. Not only does the object (as a whole) need to be transformed, but all vertices (then faces) must be transformed. It is the combination of all the vertices being transformed that cause the object to be transformed. Below is an example of the basic declaration of the vertices (as vectors) of a cube.

```
Vertex vertices[] =
{
      D3DXVECTOR3(-1.0f, -1.0f, -1.0f),
      D3DXVECTOR3(-1.0f, +1.0f, -1.0f),
      D3DXVECTOR3(+1.0f, +1.0f, -1.0f),
      D3DXVECTOR3(+1.0f, -1.0f, -1.0f),
      D3DXVECTOR3(-1.0f, -1.0f, +1.0f),
      D3DXVECTOR3(-1.0f, +1.0f, +1.0f),
      D3DXVECTOR3(+1.0f, +1.0f, +1.0f),
      D3DXVECTOR3(+1.0f, -1.0f, +1.0f),
};
```

This clearly shows each vertex that must be created by including the coordinates on each axis. All transformations are handled in the form of matrices. Depending on the transformation, a matrix may need to be multiplied by multiple other matrices. "With three dimensional graphics, a four by four matrix can hold everything you want to know about the transformation, including the angle of the viewing frustum (i.e. whether or not the camera is fish-eyed) and the way the perspective is calculated, as well as the translation, scaling and rotation required to render the scene." (Carl Muller, "Gamasutra - Mathematics In Videogames", 1997). When put into a matrix, an additional value is added to indicate if it is a position in space (x, y, z, 1) or a direction (x, y, z, 0). To translate the first vertex (-1, -1, -1, 1) 5 units in the x and z axes, it would be calculated like this:

$$\begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1*-1 & 0*-1 & 0*-1 & 5*1 \\ 0*-1 & 1*-1 & 0*-1 & 0*1 \\ 0*-1 & 0*-1 & 1*-1 & 5*1 \\ 0*-1 & 0*-1 & 0*-1 & 1*1 \end{bmatrix} = \begin{bmatrix} -1+0+0+5 \\ 0+-1+0+0 \\ 0+0+-1+5 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \\ 4 \\ 1 \end{bmatrix}$$

This results in the first vertex moving from (-1, -1, -1, 1) to (4, -1, 4, 1), and by just looking at the two, it very clearly shows that it has moved 5 in both the x and z axes. However, as more vertices (then faces, then objects) are added, the number of calculations that must be done per each frame increases exponentially. While this may seem a bit extreme, the newer graphics cards available are designed specifically to handle these processes. With every release,

the technology that handles 3D graphics is making leaps and bounds forward, meaning that future games will become even more complex, and at the same time also more realistic and believable. Math will always be a huge factor in video games, and when technology improves, it means that more complex functions will need to be processed quickly and efficiently.

Resources

"Math For Video Game Making (Or: Will I Need to Use Calculus?)." *HobbyGameDev*. N.p., 27 Feb. 2010. Web. 1 Nov. 2013. <http://www.hobbygamedev.com/articles/vol11/math-for-videogame-making-or-will-i-need-to-use-calculus/>.

Muller, Carl. "Gamasutra - Mathematics In Videogames." *Mathematics In Videogames*. Gamasutra, 1 June 1997. Web. 1 Nov. 2013. <http://www.gamasutra.com/view/feature/3197/mathematics_in_videogames.php>.

C., J., & M., S., & M., S.. (personal interviews, November 01, 2013).