# Interactive Billsbury Doughboi

**By Alan Haug**

## Abstract:

People get bored on their computers, so why not give them some simple entertainment? The interactive Billsbury Doughboi is a lightweight solution to this problem; it is fun to play with and not too in depth. The program is achieved through the use of a hierarchical figure animated with forward kinematics. This paper describes the process and techniques of creating an interactive Billsbury Doughboi, which may be applied to other forms of interactive entertainment.

## Introduction:

This work allows the user to observe a life-like humanoid figure. Furthermore, the user may interact with the figure by clicking on different body parts and watching how it would respond, perhaps if it were a real living creature. The figure is inspired by the Pillsbury Doughboy, therefore it has the signature tummy tickle move that the little guy has become so well known for. It also shares the same face as the trademark pastry mascot; this can bring a familiarity to the user so they are more comfortable with my program.

The Doughboi responds to mouse clicks in 6 different locations, and has an idle animation loop for when there is no click. The clickable locations are: left arm, right arm, left leg, right leg, head, stomach/body. A click registers an animation, but only one can play through at a time, some animations may interrupt others, and some will wait for others to finish before they begin.

This paper is meant to describe the process I went through to create the Doughboi program, and how I did it. It shall also describe the resulting interactivity and how it may be applied and extended further in the realm of interactive digital media. As a game designer, I see great potential with life like responsive humanoid figures like this for game applications.

## Process & The Gritty Details:

I began by drawing a simple body with OpenGL primitives (little did I know, I would end up just keeping it as the final figure). If I spent more time and had a longer period to work on it, I would probably have eventually translated everything into a model which looked more like the real Doughboy instead of a boxy robot dude. After the body, I then just tried to make it breath by increasing/decreasing the body size (dependent on max and min sizes as well as an inhaling Boolean so I knew which direction to go) and adjusting the arms. This looked well enough for a basic idle state. As I started to plan more and go further, I realized I would need more structure to the code implementation (as is usual with programs). So I set off to create the doughboi class and have it contain all the data as well as drawing and animating methods, which would then be called in the main display function instead of animating them in the function.

At this point I had basic movement for each body part, now I needed to have it triggered by the user. I used the picking name stack to label each body part, and I pushed them on the stack while calling the corresponding draw method from my doughboy object. After a click on some body part, it would just wiggle back and forth for a few seconds, and then the figure would return to its idle breathing state. The basic idea had been accomplished, but the polish and guts had to still be implemented: the animations. I've done very little computer animation, especially with multiple axis joints rotating in a natural human like nature. So I was ready to dive in and learn through trial and error. My goal was to do an animation in steps: first get limb to position A, then, once all angles are set, go on toward position B, and on. Finally, I wanted to end them moving back towards their original states like the idle animation.

I think I should take a step back and describe how the animations are triggered and ensure their completion before I go into the detail of how they are actually implemented to move the figure. In main.cpp, a click on a body part sets a flag telling the program that it was clicked. In the display function there are checks for each flag being set. Inside each of those checks, there is another check for the Boolean return value of the corresponding body part's animation. The animation function will return true if it is still in the process of going through the animation, and return false once it completes the animation. If the function returns false (no need to move that limb anymore) the click flag in main.cpp is reset to false so it will no longer call the animate function during future iterations of the display function until it is clicked on again.

Now I will describe my journey to creating useful animation functions and how they work. I went through a few ideas and trial implementations for playing the necessary animations of the figure. However, many of them would only play part way through (stop premature because they were based solely on a timer), or they would loop forever, or just get stuck at one of the transition points. I finally came upon the approach where I use the timer more as a progress bar. The first time the animation is called, a timer is set to 0 or its max time (depending on the function). Then each time the animate function is called in display, it checks if a certain criteria has been met. Once it has been met, the timer is incremented (or decremented), allowing the function to go to the next step. For example, the figure must first rotate his arms so his hands are up before they can rub his belly, otherwise it would look unnatural because they would go inside his chest cavity then pop out and rub it. After the animation reaches its final stage, all the affected joint angles are returned to their initial/idle positions and the function returns false (so the main program knows not to call it again until the body part is clicked on). As I got further into my work I realized some steps in the animation would need substeps. For the animation steps, I used switch statements based on the timer, and the inner substeps relied on a done int, which would be incremented when one part was complete. If done reached the specified limit, then the animation could go to the next time level (next case). This turned out to be very effective and rather intuitive after some further exposure coding with it. Here is an example of how the right leg is moved using timers, a switch statement, and a done variable:

```
bool doughboi::move_Rleg(){
	int done = 0;
	switch(Rleg_timer){
		case 0:
			if(RhipAng < 10)
				RhipAng += 1.5;
			else
				done += 1;
			if(RkneeAng > -110)
				RkneeAng -= 2;
			else done += 1;
			if(done == 2) ++Rleg_timer;
			break;
		case 1:
			if(RkneeAng < 0)
				RkneeAng += 3.5;
			else ++Rleg_timer;
			break;
		case 2:
			if(RkneeAng > initRkneeAng)
				RkneeAng -= 1;
			else ++done;
			if(RhipAng > initRhipAng)
				RhipAng -= 1.5;
			else ++done;
			if(done == 2) ++Rleg_timer;
			break;
		case 3:
			RhipAng = initRhipAng;
			RkneeAng = initRkneeAng;
			return false;
			break;
	}
	return true;
}
```

Moving the limbs based on their angles and not their endpoint positions was me using forward kinematics. I guess it makes it harder to animate with key frames, or other simple user interfaced methods, but it was easier to calculate.

In order to give the figure more life-like qualities, it had to have a pseudo will of its own. I achieved this with the breathing animation as well as random movements for the head. It has 3 options: turn side to side, turn up or down, look forward. When most of the body animations play I have it look forward, as if it were responding to an actual being poking it (user clicking mouse). Otherwise, a random number determines which way it will look; then it rests at that position, contemplating its destiny.

## Moving the Limbs using OpenGL matrix stack:

All of the figure's draw methods utilized the GL matrix stack to keep body coordinates in relation with their parent body parts. For example, the leg is translated and rotated based on the body's current matrix frame position. While not immediately obvious, all the glTransformations contained variables which

were changed during the animation methods allowing me to bend the elbow, or swing the head around. Once a body part reached the last node child, i.e. a hand or foot, the matrices were then popped of the stack accordingly so other body parts would not get affected. That was how I achieved my hierarchical structure for the figure body.

## Results:

The final figure looks a lot less than the actual Pillsbury Doughboy than I had hoped for or intended. However, I think its metallic shininess and jagged edges give it character, as if it were a robot version of the pastry icon. I am very pleased with the tummy tickle animation; that was the most important one to me, and I feel like I nailed it. Luckily I had many old and new commercials to draw from on youtube so I could study the behavioral animation of the Pillsbury Doughboy and apply it toward my own figure.

I pasted a texture of the original Doughboy's face onto my creation. That completed the aesthetic aspect of my project. If the user clicks on his tummy in succession, it looks like the guy is dancing, and he looks so happy while he's doing it too.

Sometimes the animations will slow down, other times the program runs smoothly for as long as I have it open. I'm not sure why, but that is something to be aware of while it is open. I think having other programs closed while mine is open helps it run smoother.

## Conclusion:

The end? I'm not entirely sure what else to mention in this paper. I didn't have to do much research since it wasn't physically based, or dependent on any super specific animation algorithms. Most of the fine tuning was done with trial and error. I would try out some movements, and speeds, if they didn't look right, then I would tweak the numbers until it looked right to me. In the future I want to be able to write programs that can do this sort of interactive animation with actual 3D models instead of a bunch of blocky OpenGL shapes.

## Related Work:

Processing monsters [1] are mini programs built with the Processing language. They are submitted to a website and they must be black and white and respond to the mouse. In a way, my Doughboi is the next evolutionary step beyond the Processing monster.

## References:

[1] the Processing Monsters community: http://www.rmx.cz/monsters/

- Tutorial on picking http://www.lighthouse3d.com/opengl/picking/

-Tutorial on texturing http://www.nullterminator.net/gltexture.html

- Data for animating the figure in a natural way to his character
http://www.youtube.com/results?search_query=pillsbury+doughboy&aq=f

- Other opengl information http://nehe.gamedev.net