

SNOOD-A-MOVE  
© 2001 by Ben Sibelman  
C++ console application

This game has a cross between the feel of the classic game Bust-a-Move and a variant I once played called Snood. I modified it slightly from its original form to ensure that the ASCII animations are visible on today's faster computers.

One challenge I had to work with was the non-event-driven user interface, which has to pause program execution to wait for a response from the user. My solution was to update the countdown clock while the user is moving the "arrow" back and forth. You will also notice that the collision algorithm behaves strangely at times (it resembles Snood in that respect).

Controls: left and right arrows to target, spacebar to fire a "marble," Q to quit

MOUSEHUNT  
© 2002 by Ben Sibelman  
Visual Basic 6

This is a pretty straightforward implementation of the classic game Nibbles, with a twist: the "mouse" that represents your quarry can move randomly around the spacious play area, forcing the player to chase it down and corner it. Rather than relying on an internal representation of the walls drawn onscreen, I actually had the program query for the color of a pixel ahead of the snake each timestep to determine whether it will crash.

Controls: arrow keys to move, P to pause, M to toggle moving mice

WINGS version 0.1.6  
© 2007 by Ben Sibelman  
C++ and OpenGL

This game began as my Senior Project, a simplified physical simulation of gliding and flapping flight in a bird roughly the size and shape of a pigeon. The flapping mode never produced enough lift to maintain altitude, so for this game I replaced the simulation of flapping with a simple acceleration to a preset speed in the direction of flight. The gliding mode still features Bernoulli simulation with Runge-Kutta smoothing to keep the acceleration curve stable. A partial class implementation containing the code for bird movement when flapping, gliding, or falling is presented in `bird-sample.cpp`.

Obviously the graphics are extremely simple, which facilitates the basic collision handling I added recently for the sphere-and-cone trees. The handling could use some adjustment, as it's presently rather jerky. And I may have a problem with having made it too *easy* to stay airborne, even in "bounding" mode, where the bird acts as a simple projectile when not flapping. But overall, I think the play experience feels a fair amount like flying a real bird.

To make the OpenGL engine work, `glut32.dll` must be placed in your `C:\Windows\System32` folder. See next page for the controls.

<u>Keypress</u>	<u>Usage</u>
Spacebar	Hold down or press repeatedly to flap wings
Arrow keys	Pitch bird up or down or roll left or right
P key	Pause or unpause the game
+ and - keys	Zoom in or out

<u>Context menu item</u>	<u>Purpose</u>
<i>Main menu</i>	
Quit	Close the program
Reset simulation	Reset the bird to the initial position and velocity
Toggle view mode	Control whether the camera rolls when the bird banks
Invert pitch control	Set whether the Up arrow pitches the bird up or down
Toggle flight mode	Switch between flap-and-glide and bounding flight
Set flaps per second	Set how fast the bird flaps (doesn't affect flight)
Bird control help	Display help on the keyboard and mouse controls
Menu item help	Display help on the menu items
<i>Simulation menu</i>	
Adjust gravity	Set the gravity constant in meters per second squared
Adjust starting speed	Set the bird's initial speed in meters per second
Toggle lift vector display	Set whether to show the lift vector when gliding
<i>Performance menu</i>	
Adjust refresh rate	Set how often the screen attempts to redisplay
Adjust simulation rate	Set how fast the bird moves compared to realtime