GLIDE!

Player Manual and Game Specifications

By Chris Eng and Ken Rice

CSS451 Winter 2009

Player Manual

Object of Glide!

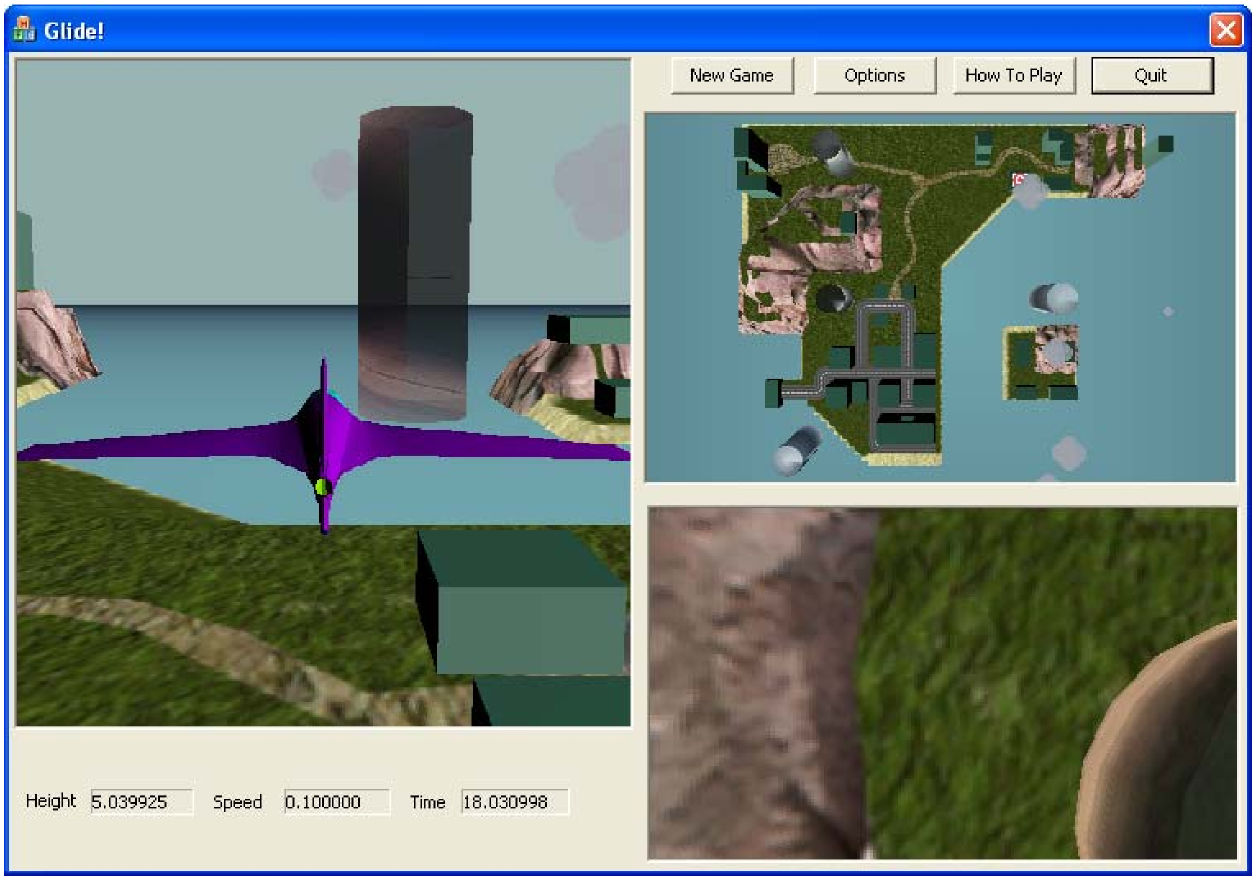
You control the flight of a glider above a small oceanic island. You may try to land the glider on the target area in the least time possible, or simply enjoy flying around the island. Either way, you should avoid colliding with terrain, buildings, and the ocean.

How to Play

1. Controls

The four arrow keys control the glider. Forward arrow points the nose down and increases the glider speed while backward points the nose up and slows the glider. The right and left arrows bank the glider in a slow turn in that respective direction. The P key pauses and un-pauses the game.

2. Game Display



Height 5.039925 Speed 0.100000 Time 18.030998

Figure 1. Game Display

Figure 1 shows the game display. The game display consists of five components: the Main View, the Map View, the Downwards View, the Glider Display, and the Game Menu.

The Main View is where primary game play occurs. You guide the glider in its flight.

The Map View shows the collection of islands from overhead. You can use the mouse to control the camera position and zoom. Left mouse dragging on the map will reposition the camera. Right mouse dragging on the map adjusts the camera zoom.

The Downward View shows the view under the glider as it flies. This view is helpful for landing on the target.

The Glider Display shows the Height and Speed of the Glider. The Time tracks how long the glider has been flying and will stop when the glider lands on the target.

The Game Menu consists of four buttons: New Game, Options, How To Play and Quit. Clicking New Game will start a new game of Glide! Clicking Options will pause the game and display the Options window where you can deactivate the Map and Downward views to improve game performance. Clicking How To Play will pause the game and display the How To Play window; here, you can see how to fly the glider and get gameplay tips. Finally, clicking Quit will close the Glider! application.

3. Gameplay Tips

You cannot flip the glider with the arrow keys. You can only rotate up to roughly 20 degrees in any given direction.

Your glider gradually loses altitude as it travels. The faster the glider is traveling, though, the slower it will lose altitude. Similarly, the slower the glider is traveling, the faster it will lose altitude. To regain altitude, you can guide the glider into one of the four *thermal updrafts* found around the island.

Try to avoid colliding with buildings, terrain, and the ocean. They will redirect your glider's path.

To correctly land on the target (thereby stopping the timer), you must bring the glider to a low enough speed atop the target. If the glider is traveling very quickly, you will not get credit for landing on the target.

If you land atop any building or terrain, you may still guide the glider around the level with the arrow keys.

Hero and World – Model and Behavior

Hero – The Glider

The glider is simply the Shusui mesh that is provided by the UWB graphic library. Pitching and rolling the glider is achieved by simply rotating the glider SceneNode about the appropriate axis of the glider's frame (x-axis and z-axis, respectively). Glider yaw is tied to the intensity of the roll, similar to actual hang gliding, and is achieved by rotating the glider SceneNode by the y-axis of the glider's frame.

The glider is always losing altitude, though the rate of decent is tied to the speed of the glider. The faster the glider goes, the greater the lift the glider achieves to counteract gravity. The glider can also gain altitude by flying through a thermal updraft.

Finally, glider collisions are handled via the tile system. If the glider's altitude is less than the altitudes registered in the tiles around it, then collision checking occurs. If the glider is found to be in an illegal space, then it is reset to a valid space.

World – Island, Water, Buildings, Target, Clouds

The island terrain is constructed using the UWBD3D_PrimitiveMeshCustom. Each control point on the mesh was then manually set to desirable elevations to create visually interesting terrain. With the terrain in place, a single texture was meticulously created and applied to the mesh to further define the terrain. To enforce collisions with the terrain, a simplistic 2D tile collision approach was implemented, made possible by modeling the terrain according to a grid pattern. Thus, each change in elevation also required registering the appropriate tile with an elevation.

The buildings were constructed of Box meshes. Like the terrain, they were placed manually for aesthetic appeal and gameplay variety. Also like the terrain, placing each building required registering the appropriate tile with an elevation in order to support collisions.

The target and ocean are rectangle meshes, while the clouds are collections of Sphere meshes. The target, ocean, and clouds were placed manually into the level for aesthetic appeal. Like the terrain and buildings, target and ocean collisions are handled via the tile system.

Glide! System Evaluation

Known Bugs

There are a few bugs remaining in the current version of Glide!

The collision detection is not perfect in a few areas of the island, particularly around the cliffs and hillsides. The collision-detection resolution is relatively broad, meaning collisions sometime occur too early or too late relative to the onscreen graphics.

When rotating the glider fully in any direction, the glider will stutter back-and-forth at the limit (roughly 20 degrees). When pitching upward or downward, this can cause unsteady speed increases or decreases.

Limitations

Glide! Was designed with ambitious goals in mind, like its 2D predecessor, *Escape!* As development wore on, it became apparent that we would not be able to reach every goal we initially aimed for. In the end, *Glide!* only lightly features a target-and-timer approach and instead focuses primarily simply flying the glider around.

Most noticeably, the glider mesh itself was intended to be an actual hang glider with an onboard pilot. However, there are very few hang glider meshes online that are free. The lone free, reasonable mesh we found was too geometrically taxing upon the game engine, forcing us to settle with the “Shusui” airplane mesh.

Early versions of the glider physics allowed for unrestricted rotations in all directions. However, this led to orientation frame issues, prompting us to heavily restrict rotation to prevent these problematic states. In addition, the physics are highly simplistic and do not emulate true reality. Speed and lift are tied directly to pitching the glider forward and back, for instance, rather than using the actual hang glider equations that tie lift with drag and gravity. Collisions, meanwhile, are calculated using a “tile model” similar to the one in *Escape!*, meaning the accuracy of some collisions are inaccurate (flying underneath an overpass, for example, is currently impossible).

Originally we planned to implement clouds, birds, and a “follow-camera” that would offer a secondary viewing option in the Downward View space. However, only a basic version of clouds was implemented; they orbit the world center, though this implementation seems adequate.

The thermal updrafts currently use a basic cylinder mesh to denote areas where the player can gain altitude. The mesh is not aesthetically pleasing to the eye, but it is passable when made semi-transparent.

Glide! does not feature any sound due to synchronization issues with Window’s basic PlaySound functionality. It could only play one sound at a time and is not easily compatible with playing sounds upon collisions. Thus, only pausing and unpausing the game plays a tone.

Finally, only simplistic lighting and textures were used in the game. Thus, true shadows do not appear on the island and the texturing of the terrain is slightly off in places. Additionally, the sky and ocean were implemented using only basic effects; the sky was implemented by simply clearing the drawing buffer to sky blue, while the ocean was implemented by placing a large primitive rectangle accordingly.

Possible Extensions in Future Versions

In future updates of *Glide!*, all initial gameplay features should be implemented. This includes birds, more appealing thermal updraft and clouds, the “follow-camera,” engaging audio, an actual hang glider mesh, a “ring-path” race mode, and a deeper target mode.

Future updates should also include a more realistic glider physics model, one that calculates lift versus gravity and drag. Additionally, collisions should greatly impact the glider speed to encourage players to fly around obstacles rather than into them. Additionally, a “walking mode” should be included to appear more realistic when the glider is grounded.

Improvements to the current tile-collision model should allow the glider to fly underneath overpasses, though a more robust glider-mesh intersection approach would be even more accurate than using tiles while also achieving flight underneath buildings.