Foreign Function Interfaces, OpenGL and Game Programming in Haskell

Hugh Rayner
Structure

• Foreign Function Interfaces, their advantages and disadvantages

• Haskell OpenGL

• Game programming in Haskell

• GLider
Foreign Function Interfaces

• Allow access to functions not in Haskell.

• Allow globals that aren’t passed around.

• Need to be kept up-to-date with other language libraries.

• Can be subject to change.

• Can make error tracking difficult.
Game Programming in Haskell

• IORefs used to store game state.

• Purely functional functions and Monadic functions.

• No global state - state must be passed explicitly - encourages use of a smaller data structure or a single “state” type.

• Dynamic and static state.

• Game physics - Realism vs Playability
My Project - GLider

• 3d remake of an old game - fly a paper aeroplane through a house, using air vents for lift, avoiding obstacles.

• Physics - a more unrealistic but more playable model used.
GLider - How Haskell Helped

- GLider uses a lot of matrix operations. It is far, far easier to write matrix operations in Haskell than C (even if they don’t run as fast).

- Haskell tends to be more terse than C, code easier to maintain.

- Defining level in a haskell file is simple - can easily define objects in terms of other objects.

- Not much data needs to be stored in GLider.
GLider - How Haskell Helped

• Strong typing reduced number of errors.

• Higher order functions such as preservingMatrix replacing pairs of functions reduced errors like an unpopped matrix.

• Vector function arguments meant neater code
GLider - How Haskell Hindered

• Relatively difficult to transfer information from control callbacks to game physics.

• Inconsistencies and gaps in Haskell OpenGL.

• Incentive to keep number of state variables low. Difficult to reassign part of a data structure.

• The function bindings in Haskell are significantly different to C.
Conclusion

• Haskell is a mixed blessing for graphics and/or game programming.

• For GLider, a small number of state variables were required. This meant Haskell helped.

• For most games requiring a large number of actors the benefits of Haskell would probably be outweighed by the need to retain a large number of IORefs.