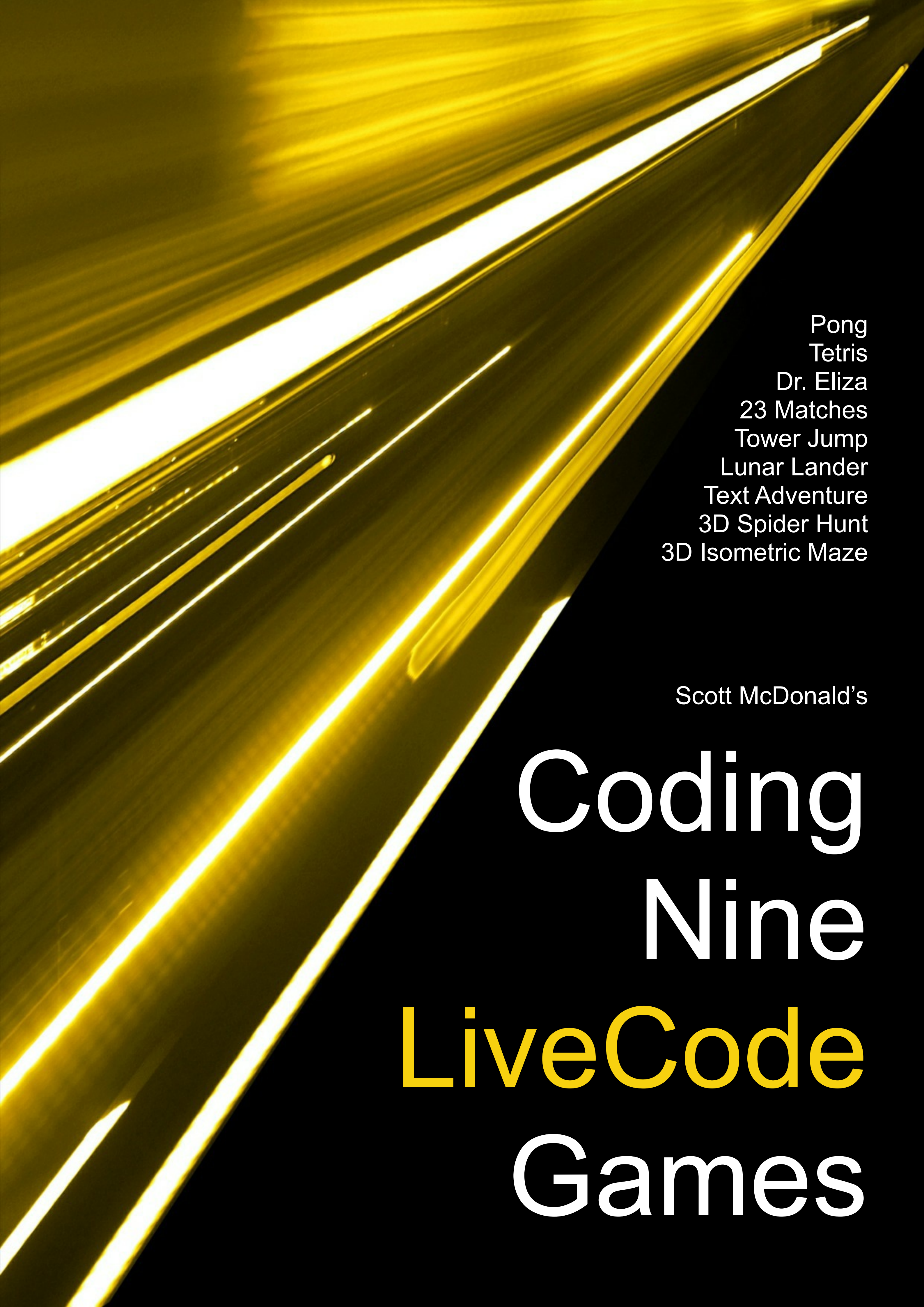


Pong
Tetris
Dr. Eliza
23 Matches
Tower Jump
Lunar Lander
Text Adventure
3D Spider Hunt
3D Isometric Maze

Scott McDonald's

Coding Nine LiveCode Games

Contents

Introduction

LiveCode Games

Coding Nine LiveCode Games has content from the LiveCode Game Developer plus 3 new LiveCode games:

- Tower Jump
- Doctor Eliza
- Lunar Lander

Plus access to the full source code of Scott's 3D raycasting demo:

- Spider Hunt

Unlike the other games in this book, Spider Hunt is not in the public domain and is only available to owners of this book. See the *Where To Get The Code* chapter at the end for your unique download code for this interesting and innovative LiveCode stack.

If you already visit http://livecodegamedeveloper.com you know why LiveCode and Games go together.

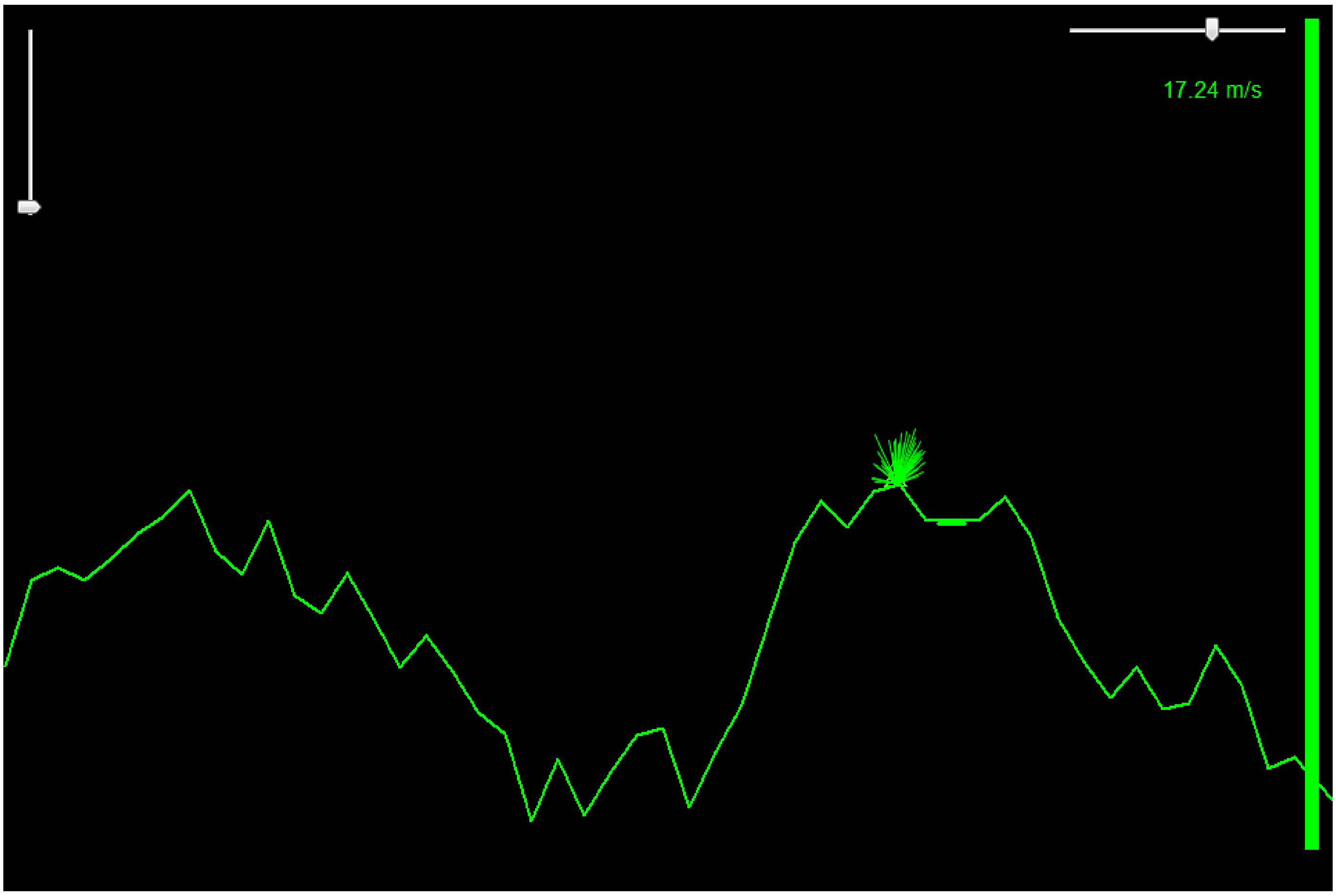
With LiveCode you can make a game with a fraction of the effort of other languages.

With the exception of Spider Hunt, all the games in this book are less than 500 lines long, and can be coded in a weekend

By playing the games and examining the techniques used, you will discover ways of creating interesting games with a minimum of code, that are fun to play. Each chapter includes suggested ways to enhance the existing code to make the game even better. How will you improve each game, and make it your own unique version of a classic game?

Happy LiveCoding!

Scott McDonald



Oops! Yet another crash near the landing site.

Game 6

Lunar Lander

Another true classic. This was around before personal computers were invented. The very first versions were text only games that printed as text your height and velocity above the moon. Then you specified the amount of fuel you wanted to burn. The aim being make sure the velocity is small enough when reaching the surface so the LEM wouldn't crash. The mathematics that powered it was reasonably easy to understand and with a bit of practise you could master it with a safe landing every time.

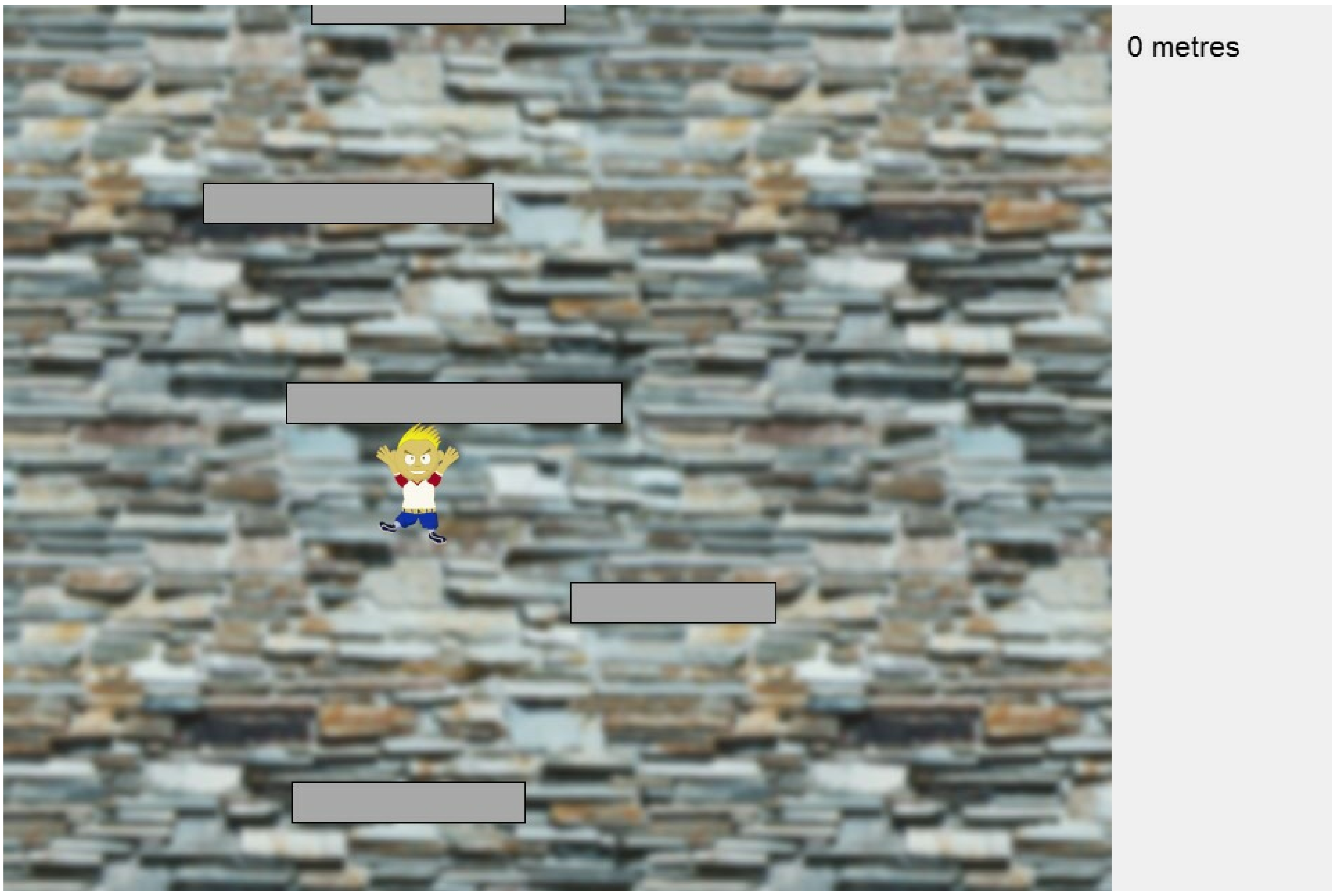
The game here is a graphical Lunar Lander inspired on later versions from the 1970's that introduced vector styled graphics and 2 dimensions of movement.

In this game you will see:

- A random moonscape made with a single graphic object
- Scroll bars used to indicate a value and allow control by touch
- A simple animation for a crash landing
- Handling of more than one key pressed at the same time
- Multiple images used to animate the LEM

Below is the StartGame handler. This handler is called from the openCard handler and calls commands to initialise everything. Then ShowInstructions displays important information to the player. Originally this game was developed with a touch device in mind so while the instructions refer to keys, the game can be played by touch alone by using the scroll bars to control the thrust.

```
command StartGame
  lock screen
  CancelPendingMessage "GameLoop"
  InitializeLEM
  InitializeSurface
  InitializeFuelThrust
  InitializeStatus
  ShowInstructions
  unlock screen
```



How many metres can Anime Boy jump?

Game 7

Tower Jump

Another arcade style game, which I call Tower Jump. Before the indie game scene really took off and became the cool label for individual and small teams of independent game developers, there was Free Lunch Design. In 2001, Johan Peitz and team made Icy Tower, a very fun addictive free game. Which according to Wikipedia is based on an earlier Linux game named Xjump or Falling Tower.

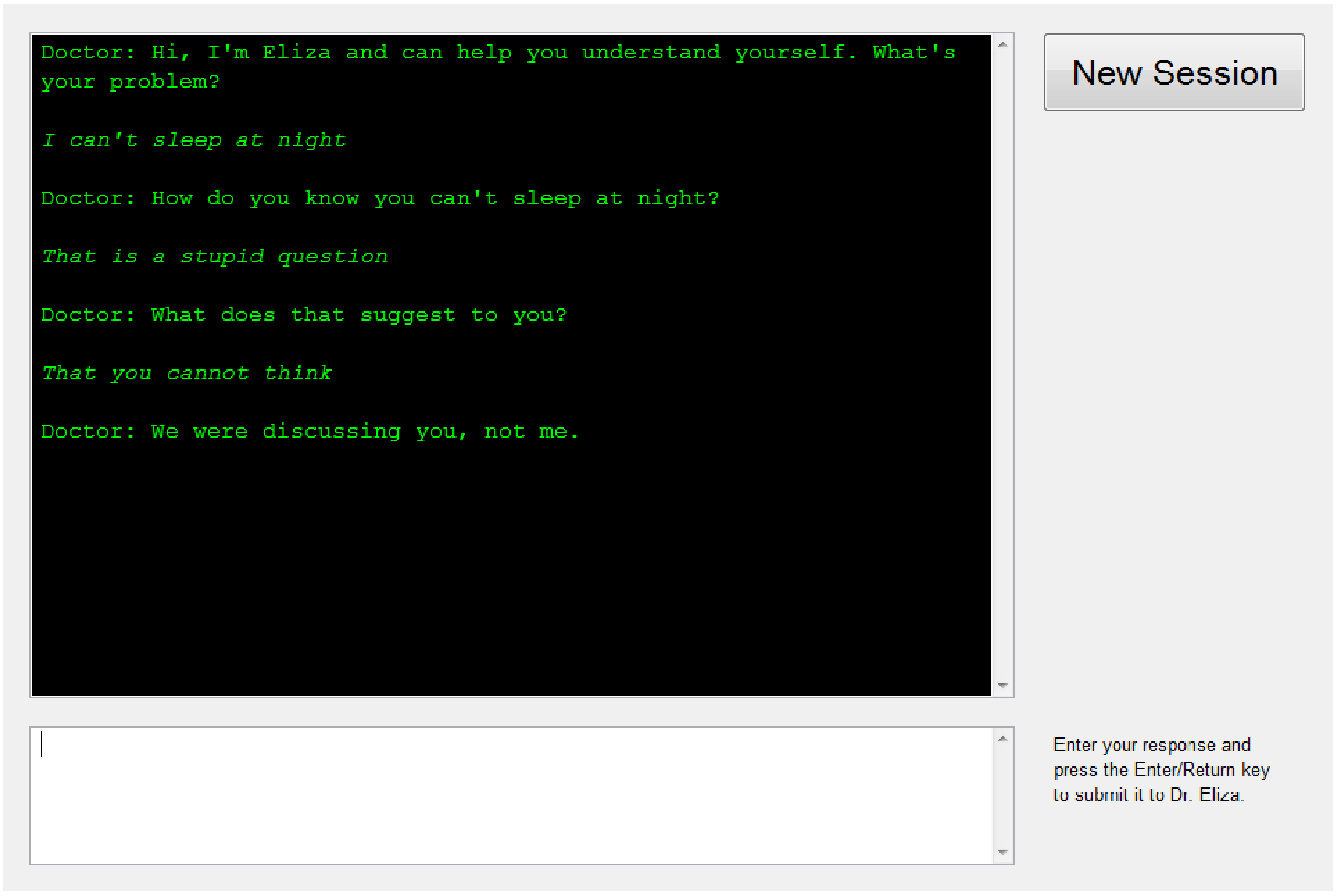
In this game you will see:

- An endless scrolling background
- A LiveCode scrolling group
- An animated character
- Collision detection with one way platforms

The StartGame handler calls ResetGame, InitializeGraphics, InitializeHero, InitializePlatforms and starts the game loop. ResetGame and InitializeGraphics deletes and re-creates graphics every time you run the game. This is done because during development it is not uncommon to end up with extra copies of the graphics. So to keep the stack tidy, deleting and making fresh graphics is done each time you play. But in a polished game, you would reorganise the code to minimise the number of objects created at runtime to speed up the loading of the game.

```
on StartGame
  ResetGame
  InitializeGraphics
  InitializePlatforms
  InitializeHero

  put 0 into sGameLoopCounter
  put false into sGameOver
  hide field "labeGameOver"
  GameLoop
end StartGame
```



```
Doctor: Hi, I'm Eliza and can help you understand yourself. What's
your problem?

I can't sleep at night

Doctor: How do you know you can't sleep at night?

That is a stupid question

Doctor: What does that suggest to you?

That you cannot think

Doctor: We were discussing you, not me.
```

New Session
Enter your response and press the Enter/Return key to submit it to Dr. Eliza.

Difficulty sleeping? Let the doctor help you.

Game 8

Doctor Eliza

Time for some more artificial intelligence (stupidity?), or A.I. for short. The program this time isn't really a game. Way, way back in the 1960's one of the first chatterbot programs was written. Called ELIZA it used simple text processing tricks in an attempt to carry its side of a human conversation. The text processing tricks are so simple that calling it artificial intelligence is laughable, but it can produce uncanny responses, which encourages the human user to give it more credit than is due. At the time it created quite a stir.

The original ELIZA could be programmed for different styles of conversations, with the Doctor style the most successful. Playing the role of a non-directive pyschologist/counsellor meant the program could produce vague and reflective comments, that could seem plausibly human, sometimes. Which resulted in some users asking to be left alone during their personal conversations with the "Doctor". (The version here is not as complex, and it is difficult to imagine this happening with it.)

In this game you will see:

- Sorting a list with one line of code
- How to scroll to the bottom of a text field
- The power of using a custom item delimiter
- A danger of Variable Preservation in the IDE

Because the code is so short, less than 200 lines including the text of all possible replies, I was struggling to find interesting items for the list above. This brevity is possible because of the high level of the LiveCode language, and so this chapter is shorter than the rest because LiveCode made writing the code way too easy. (Which is good.)

StartGame initializes the list of replies and the opposites that sometimes transform the text typed by the user into a form ready to be reflected back in the Doctor's reply. The memoOutput Text Field is filled with the introductory text and the keyboard focus set to memoInput, ready for the user to begin typing.