Intelligent Game Designer
Group Report

Team Juliet

February 25, 2016

Introduction

In this document we reflect upon the successes and failures within our project, and highlight the contributions made by each team member. Please note that bullet points highlighted in **bold** indicate requirements that were initially specified in our project criteria (see section 2.4 of 'Team Juliet - Specification').

Successes

We believe the successes of our project include, but are not limited to:

- **Users can, within the level designer interface, specify the type of level they would like generated and its target difficulty.**
- **With this information, the level designer should produce a finite series of suggested level designs, that are readable by our implementation of the game.**
- **The simulated players should be able to play the game, and capture a range of abilities similar to that of human users.**
- **Levels made by the level designer should be close to the difficulty that is requested by the user.**
- Users can manually design their own levels, and then test them out by playing them themselves, or by watching simulated players tackle them. In reality, human testers may be a limited/expensive resource, so simulated players can certainly accelerate the evaluation of a given design in a real-world context.
- Users can save and load levels that they, or the level designer have created.

Failures / Improvements That Could Be Made

We believe the failures of our project include, but are not limited to:

- Whilst our software can generate reasonable level designs, it still provides no match for the artistic creativity of a human. More specifically, aspects of levels that humans can create which our software cannot:
  - Humans can design levels which require particular strategies, such as level 31 of Candy Crush Saga, in which you must create and detonate vertically-striped candies and/or colour bombs to clear the isolated jelly cells above.
  - Humans can consistently design fascinating board patterns, whereas our software produces designs as a result of a fairly arbitrary, evolutionary process which occasionally produces fascinating board patterns.
- An additional feature that could have been added to the interface would have been to display the predicted difficulty level of particular designs as you create them.
- Improving the appearance of the game whilst in play, the inclusion of sound effects, full animations and music would have been a good addition.
- The interface itself could be better looking. Although it has been recoloured and makes use of custom fonts, due to an incomplete understanding of Java Swing's methods, some small features were not successfully modified in time (for instance the colour on the sliding bars). Adding textures and background images may have also made it look more appealing.

Lessons Learnt

- Throwing more parameters at a genetic algorithm leads to an exponentially slower convergence rate - you should only include those parameters that are absolutely necessary.
- Designing game levels is an art - one in which it is hard to outperform humans.
- Scheduling tasks within a project such as this is crucial to allow all members to work effectively and complete in time (ensuring any code that acts as a groundwork for other features is completed to a basic level as soon as possible).

Contributions

The contributions of each team member to the project *implementation* are as follows:

Alastair

Alastair primarily worked on the fitness tests for the genetic algorithm, in particular the constraints, aesthetics, difficulty and fun fitness tests. These combine data from the simulation players and a variety of image processing algorithms. Optimizing the fitness test's fixed parameters to produce good levels and the method by which we combine the separate test's results were worked on by Alastair. He also assisted in the coding of some of the genetic algorithms features, and prototyped a more advanced design representation for the genetic algorithm, which we didn't get time to implement fully.

Artem

Artem mainly focussed on the implementation and calibration of simulated players. He and Dimitris decided to use A* search based algorithm for the more sophisticated simulated players, therefore most of the work was done in finding and optimising metric and potential functions for the algorithm to use. To separate inner workings of the simulated players from the rest of the program a manager was written by Artem which selects which player would evaluate the board based on requested ability. Artem also helped a bit with core game implementation.

Ben

Ben's role within the group was as interface designer, integrating the contributions of the other team members into a product that could be viewed and demonstrated. This included a screen from which a person could play a level, or watch one of the simulated players play. He wrote some additional features such as an interface for humans to design levels, and the capability to save, load and modify designed levels.

Ciaran

Ciaran has focussed mainly on the level designing aspect of the project, writing the classes for representing and generating levels using a genetic algorithm with Devan, and implementing mutation and crossover. He has continued to make incremental improvements to the genetic algorithm and level representations to improve performance and the quality of the levels generated, and also done some of the work on writing and tweaking the functions for evaluating the aesthetic fitness of the levels.

Devan

Devan's main contribution to the implementation of the project was within the level design process. He co-wrote the classes handling the representation of the levels, as well as the genetic algorithm used to generate candidate level designs for the user. Devan's other contributions include the classes which handle running and evaluating the performance of the simulated players, small components of the game implementation and level design unit tests.

Dimitris

Dimitris' main contribution was in the core game implementation and the development of simulated players with Artem. The work in core game was around the simulation of the actual game mechanics including the testing and documentation of the operations. Several different strategies for game metrics were tested and assessed for the A* state searching algorithm. In the first phase of development he also worked on experimental game displayer for viewing the operation of simulated players and playing the game.