Anttris – CSE 326

Chris Aikman Benji Cope Skyler Manzanares
Hugo Rivera Sean Turner

April 27, 2015

Overview

Why

The Game

Workflow

Demo

Q A

Motivation

Motivation

Wanted to make an interactive and competative game

Motivation

Wanted to make an interactive and competative game

Really enjoyed the idea

Objective

- Project Considerations

Objective

- Project Considerations
  - Size

Objective

- Project Considerations
  - Size
  - Entertainment Value

Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability

Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability
- Puzzle Game

Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability
- Puzzle Game
- Competitive Edge

Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability
- Puzzle Game
- Competitive Edge
  - Play Against A Friend

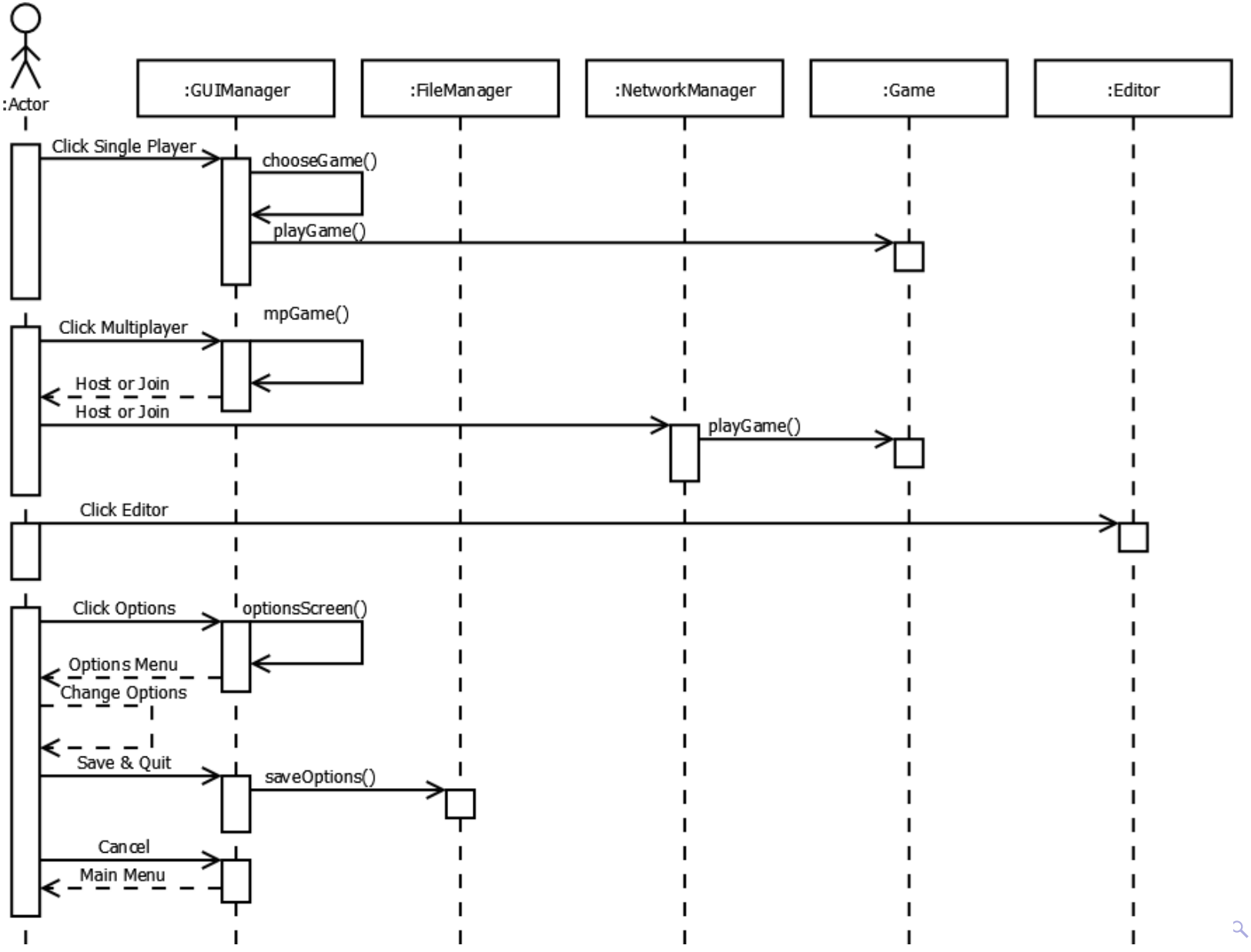
Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability
- Puzzle Game
- Competitive Edge
  - Play Against A Friend
  - See Opponent Playing

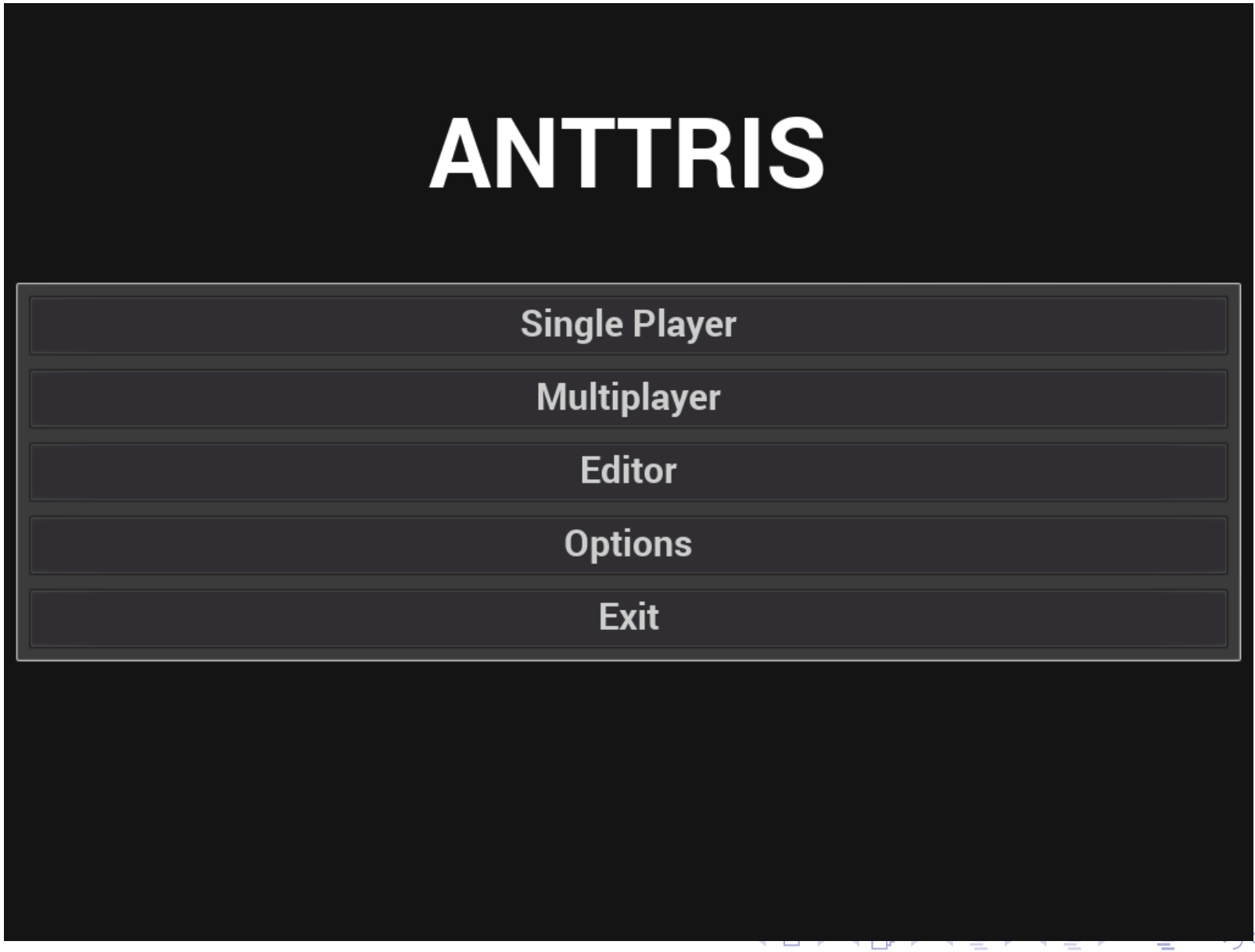
Objective

- Project Considerations
  - Size
  - Entertainment Value
  - Portability
- Puzzle Game
- Competitive Edge
  - Play Against A Friend
  - See Opponent Playing
- Custom Puzzles

GUI


:Actor
:GUIManager
:FileManager
:NetworkManager
:Game
:Editor
Click Single Player
chooseGame()
playGame()
Click Multiplayer
mpGame()
Host or Join
Host or Join
playGame()
Click Editor
Click Options
optionsScreen()
Options Menu
Change Options
Save & Quit
saveOptions()
Cancel
Main Menu

GUI



Data Manager

- Serialize Data

Data Manager

- Serialize Data
- Save / Load Options

Data Manager

- Serialize Data
- Save / Load Options
- Save / Load Puzzles

Editor

- Add/replace/remove different block types
- Load/save

Editor

- Add/replace/remove different block types
- Load/save
- Mobile friendly interface

Editor

- Add/replace/remove different block types
- Load/save
- Mobile friendly interface
- Layer management: create random, remove current (Stretch goal)

Generator

The Problem:

Generator

The Problem:

- Need to generate puzzles

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

The Solution:

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

The Solution:

- Generate all positions

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

The Solution:

- Generate all positions
- Randomize

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

The Solution:

- Generate all positions
- Randomize
- Assign based on position

Generator

The Problem:

- Need to generate puzzles
- Need to be solvable
- Need to follow the rules

The Solution:

- Generate all positions
- Randomize
- Assign based on position
- Randomize pairs

Solver

The Problem:

Solver

The Problem:

- Need to check for solutions

Solver

The Problem:

- Need to check for solutions
- Check random puzzles

Solver

The Problem:

- Need to check for solutions
- Check random puzzles
- Check puzzles from the editor

Solver

The Problem:

- Need to check for solutions
- Check random puzzles
- Check puzzles from the editor

The Solution:

Solver

The Problem:

- Need to check for solutions
- Check random puzzles
- Check puzzles from the editor

The Solution:

- Pull out pair blocks

Solver

The Problem:

- Need to check for solutions
- Check random puzzles
- Check puzzles from the editor

The Solution:

- Pull out pair blocks
- Make sure all pair blocks have a pair

Solver

The Problem:

- Need to check for solutions
- Check random puzzles
- Check puzzles from the editor

The Solution:

- Pull out pair blocks
- Make sure all pair blocks have a pair
- Check if the pair is on the same layer

Networking

- Multiplayer

Networking

- Multiplayer
- P2P

Networking

- Multiplayer
- P2P
  - Non-random Opponents

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model
- Key Events

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model
- Key Events
  - Start of Game

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model
- Key Events
  - Start of Game
  - Transform Puzzle

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model
- Key Events
  - Start of Game
  - Transform Puzzle
  - Select Blocks

Networking

- Multiplayer
- P2P
  - Non-random Opponents
  - Network Information
  - Server-Client Model
- Key Events
  - Start of Game
  - Transform Puzzle
  - Select Blocks
  - Game End

Rules

- The puzzle grid

Rules

- The puzzle grid
- Pair blocks

Rules

- The puzzle grid
- Pair blocks
- Laser blocks

Rules

- The puzzle grid
- Pair blocks
- Laser blocks
- Wild blocks

Rules

- The puzzle grid
- Pair blocks
- Laser blocks
- Wild blocks
- Score

Approach

Customer Requirements a fast, multiplatform block puzzle game with multiplayer and an editor.

System Design We based it heavily on Godot classes and previous work with prototypes (a Three.js and a Unity prototype)

Object Design GridMan, AbstractBlock, CameraControl

Implementation and Testing Gut.gd covered silly bugs

Godot

GD Script

GD Script

- Domain specific language

GD Script

- Domain specific language
- Object oriented, python-like

GD Script

- Domain specific language
- Object oriented, python-like
- Automatic concurrency, automatic resource caching

GD Script

- Domain specific language
- Object oriented, python-like
- Automatic concurrency, automatic resource caching
- Dynamic, bugs easy to introduce

GD Script

- Domain specific language
- Object oriented, python-like
- Automatic concurrency, automatic resource caching
- Dynamic, bugs easy to introduce
- Example: methods passed as (object, string) tuples

GD Script

- Domain specific language
- Object oriented, python-like
- Automatic concurrency, automatic resource caching
- Dynamic, bugs easy to introduce
- Example: methods passed as (object, string) tuples
- Files are classes

```
1  extends ‘‘Food.gd’’
2
3  func fry():
4      return self.fried()
5
6  class GrillFuel:
7      ...
```

GD Script

- Domain specific language
- Object oriented, python-like
- Automatic concurrency, automatic resource caching
- Dynamic, bugs easy to introduce
- Example: methods passed as (object, string) tuples
- Files are classes

```
1  extends ‘‘Food.gd’’
2
3  func fry():
4      return self.fried()
5
6  class GrillFuel:
7      ...
```

- Tightly integrated with all of Godot's C++ classes. Fast where it counts.

Godot

- Godot was easy to learn.

Godot

- Godot was easy to learn.
- Library consisting of 400+ classes, less so

Godot

- Godot was easy to learn.
- Library consisting of 400+ classes, less so
- Limited use of the neat GUI interface, binary files don't agree with Github

Our Workflow

Our Workflow

- We used Github as our source control.

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!
  - Which means, we didn't always adhere to the rules.

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!
  - Which means, we didn't always adhere to the rules.
- Continuous Integration

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!
  - Which means, we didn't always adhere to the rules.
- Continuous Integration
  - Travis CI. It's awesome and everyone should try it!

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!
  - Which means, we didn't always adhere to the rules.
- Continuous Integration
  - Travis CI. It's awesome and everyone should try it!
  - Not to mention free for open source projects.

Our Workflow

- We used Github as our source control.
- To facilitate an agile workflow we used Zenhub.
  - Zenhub integrates a Scrum board right in Github.
  - We set out to use certain parts of Scrum, but we're students!
  - Which means, we didn't always adhere to the rules.
- Continuous Integration
  - Travis CI. It's awesome and everyone should try it!
  - Not to mention free for open source projects.
  - Godot has a headless GNU/Linux server project which turned out to be perfect for this, with some scripting (more on this shortly)

Workflow, Continued

Workflow, Continued

- Branches

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.
  - This facilites easy code reviews before things get broken!

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.
  - This facilites easy code reviews before things get broken!
- Unit testing

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.
  - This facilites easy code reviews before things get broken!
- Unit testing
  - We used the [G]odot [U]nit [T]esting framework, aka GUT.

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.
  - This facilites easy code reviews before things get broken!
- Unit testing
  - We used the [G]odot [U]nit [T]esting framework, aka GUT.
  - This works nicely with Travis, thanks to a Python script Hugo wrote to catch return values and report on unit test results.

Workflow, Continued

- Branches
  - We used branches and pull requests to merge changes.
  - This facilites easy code reviews before things get broken!
- Unit testing
  - We used the [G]odot [U]nit [T]esting framework, aka GUT.
  - This works nicely with Travis, thanks to a Python script Hugo wrote to catch return values and report on unit test results.
  - This project is hosted on Bitbucket at https://bitbucket.org/bitwes/gut/overview.

Project Pace

Project Pace

Planning

Project Pace

Planning

Game Rules

Project Pace

Planning

Game Rules

Puzzle and Blocks and Networking

Project Pace

Planning

Game Rules

Puzzle and Blocks and Networking

Testing

What we learned

What we learned

- Nothing.

What we learned

- Power of Starting Early

What we learned

- Power of Starting Early
- Risk of Distributed Workload

What we learned

- Power of Starting Early
- Risk of Distributed Workload
- Applicability of Agile Methods

Demo