the gamedesigninitiative at cornell university

Lecture 8

Prototyping

What is a Prototype?

- An *incomplete* model of your product
  - Implements small subset of the final features
  - Features chosen are the most important **now**
- Prototype helps you visualize **gameplay**
  - Way for you to test a new game mechanic
  - Allows you to tune mechanic parameters
  - Can also test (some) user interfaces

What is a Prototype?

- A prototype helps you visualize **subsystems**
  - Custom lighting algorithms
  - Custom physics engine
  - Network communication layer
- Fits naturally with the SCRUM sprint
  - Identify the core mechanic/subsystem to test
  - Develop subsystem separately in sprint
  - If successful, integrate into main code

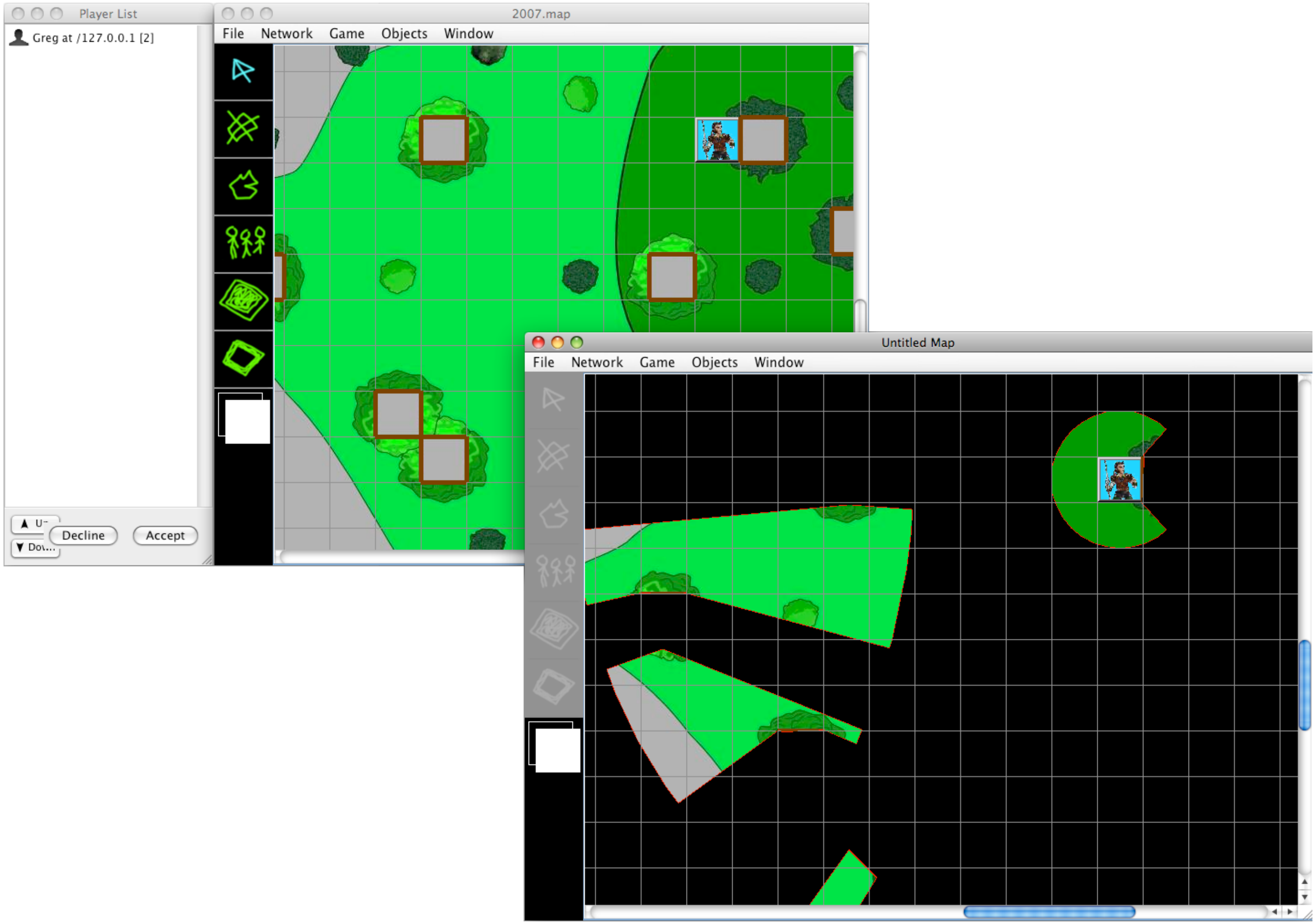
Types of Prototypes

- **Throwaway prototyping**
  - Prototype will be discarded after use
  - Often created with middleware/prototyping tool
  - Useful for **gameplay prototype**
- **Evolutionary Prototyping**
  - Robust prototype that is refined over time
  - Code eventually integrated into final product
  - Useful for your **technical prototype**

Case Study: Playing Fields

- Computer map aid for playing D&D
  - Provides a map grid for moving tokens about
  - Tools for creating tokens and images
  - Network support for a DM with many players
  - Intelligently obscures player visibility
- **Motivation**: lessen player "metagaming"
  - Physical map displays too much information
  - Playing over a network is a secondary concern

Case Study: Playing Fields



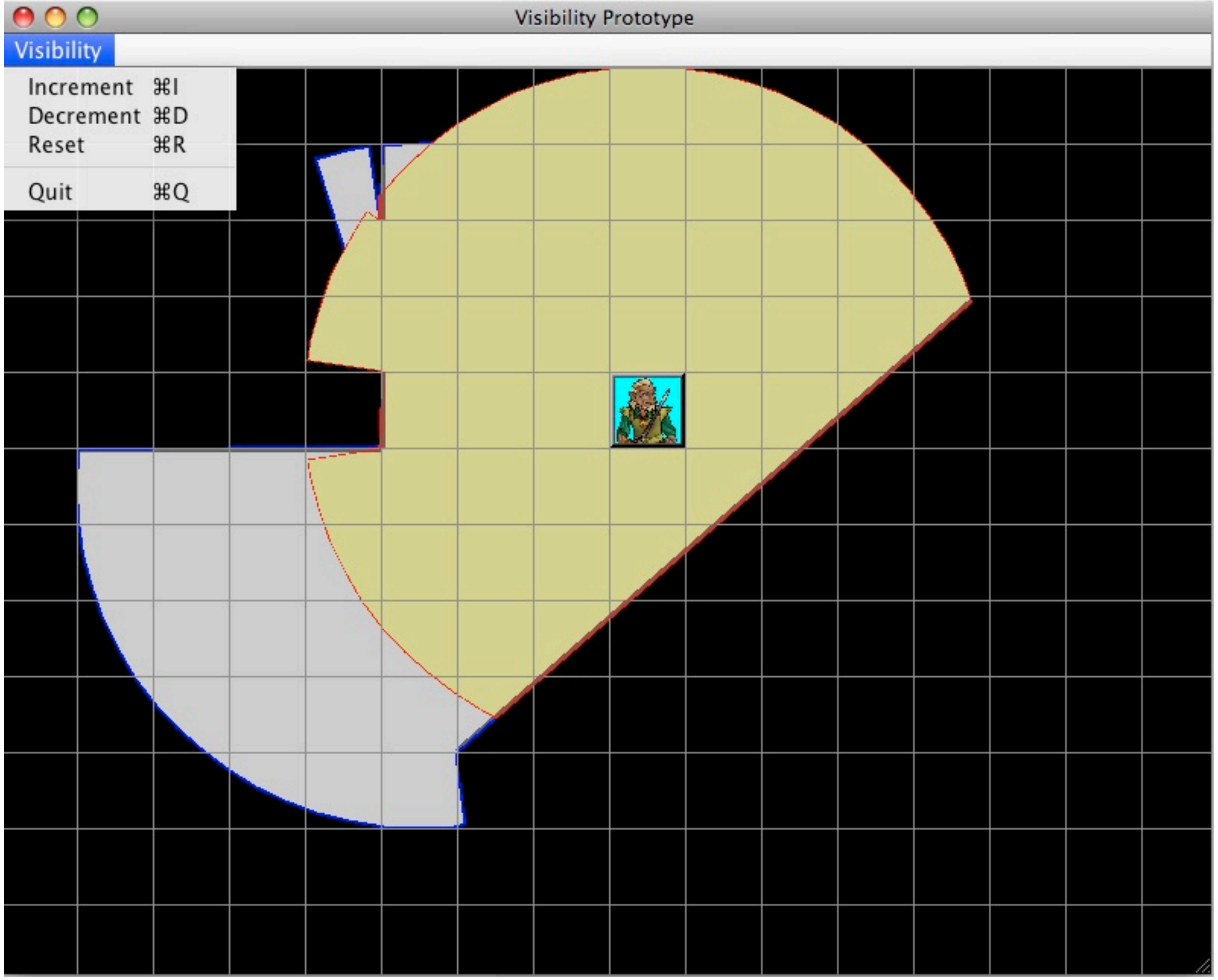
Gameplay Prototypes

- Focus on core mechanic (e.g. verb/interaction)
  - May want more than one for emergent behavior
  - But no more than 2 or 3 mechanics
  - Keep challenges very, very simple
- Prototype should allow *tuning on the fly*
  - Requiring a recompile to tune is inefficient
  - Use menus/input fields/keyboard commands
  - But do not make the UI too complicated either

Prototyping Playing Fields

- What are the core mechanics?
  - Moving a token about a grid
  - Using obstacles to block visibility
- Focuses on **visibility** and **user control**
  - Use a single token with fixed obstructions
  - Do not support network play
  - Do not worry about invalid moves
- Visibility distance is a *tunable* parameter

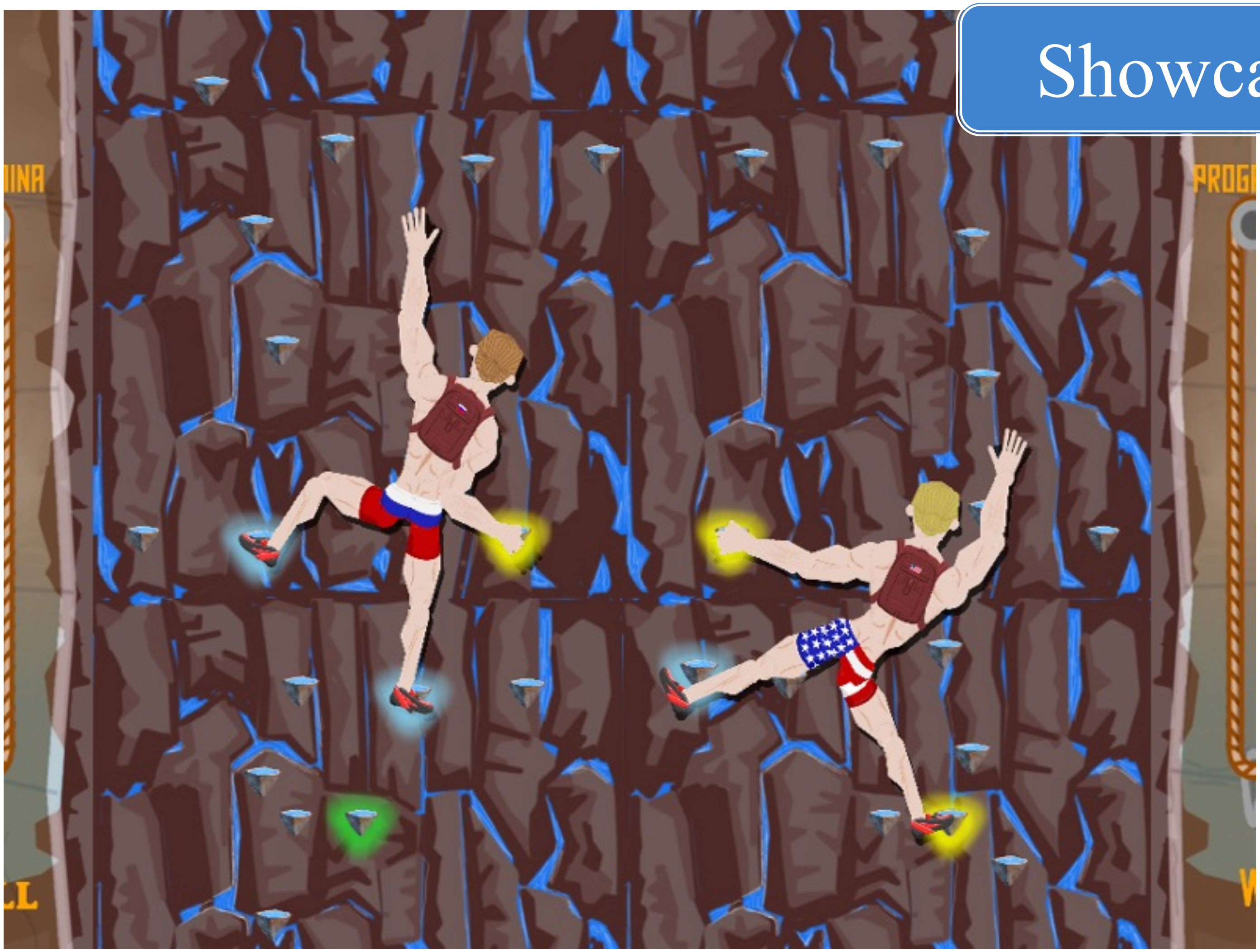
Playing Fields Prototype



Prototype: Lessons Learned

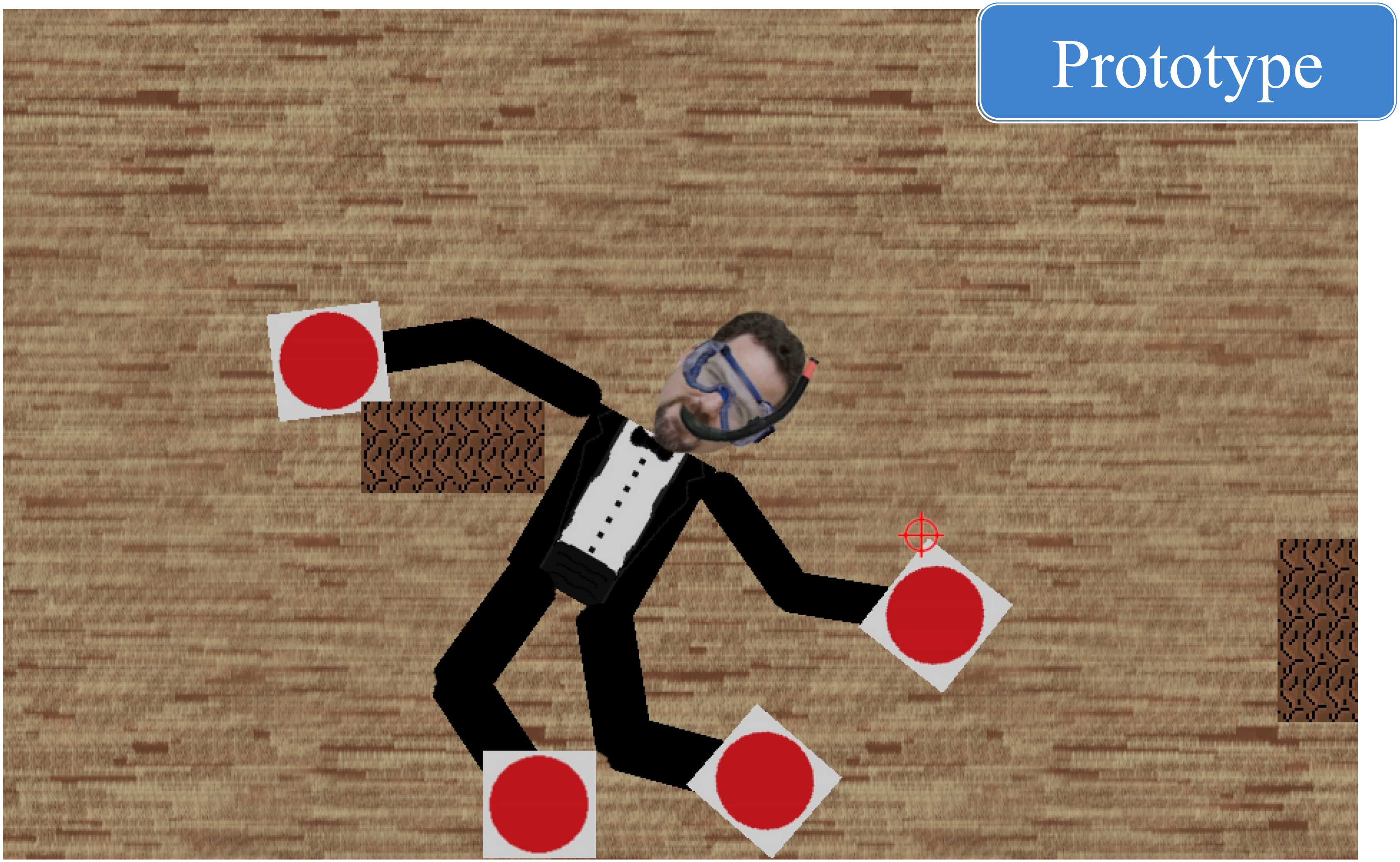
- Algorithm makes it difficult to see walls
  - May want unseen area a color other than black
  - May want to “fudge the edge of the boundary”
- Update algorithm does not support “strafing”
  - Vision is updated at start and beginning of move
  - Nothing “in between” is counted (e.g. alleys)
- Spacing of 50 pixels is optimal for viewing

3152 Example: Mount Sputnick



Showcase

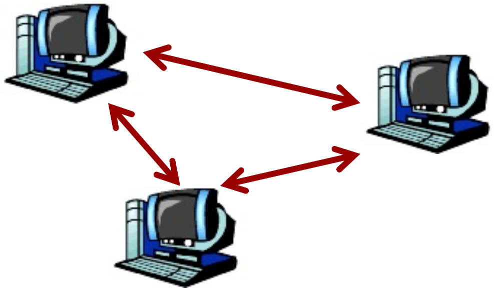
3152 Example: Mount Sputnick



Prototype

Technical Prototyping

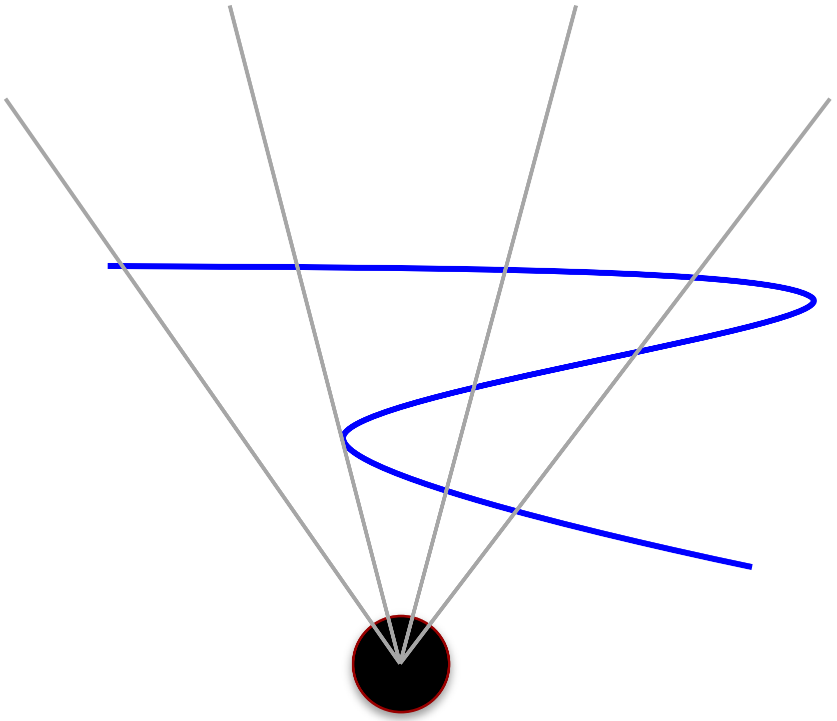
- Technical prototypes used for *subsystems*
  - Custom lighting algorithms
  - Custom physics engine
  - Network communication layer



- **Goal**: inspect inner workings of software
  - Features might be "invisible" in normal game
  - Specialized interface to visualize process
- **Not-a-Goal**: Make something fun

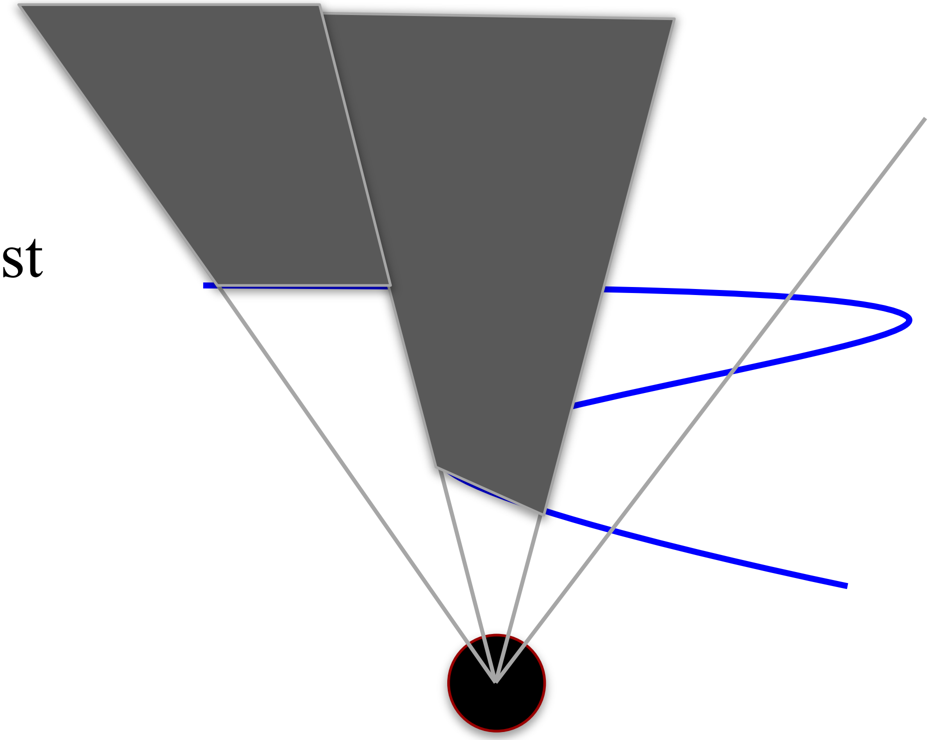
Case Study: Shadows and Lighting

- Recall gameplay prototype
  - Discrete shadows are easy
  - But had many problems
- Want something more robust
  - Continuously movement
  - Curved wall edges
  - Self-intersecting shadows
- Different features to test
  - Moving an avatar
  - Reconfiguring the wall

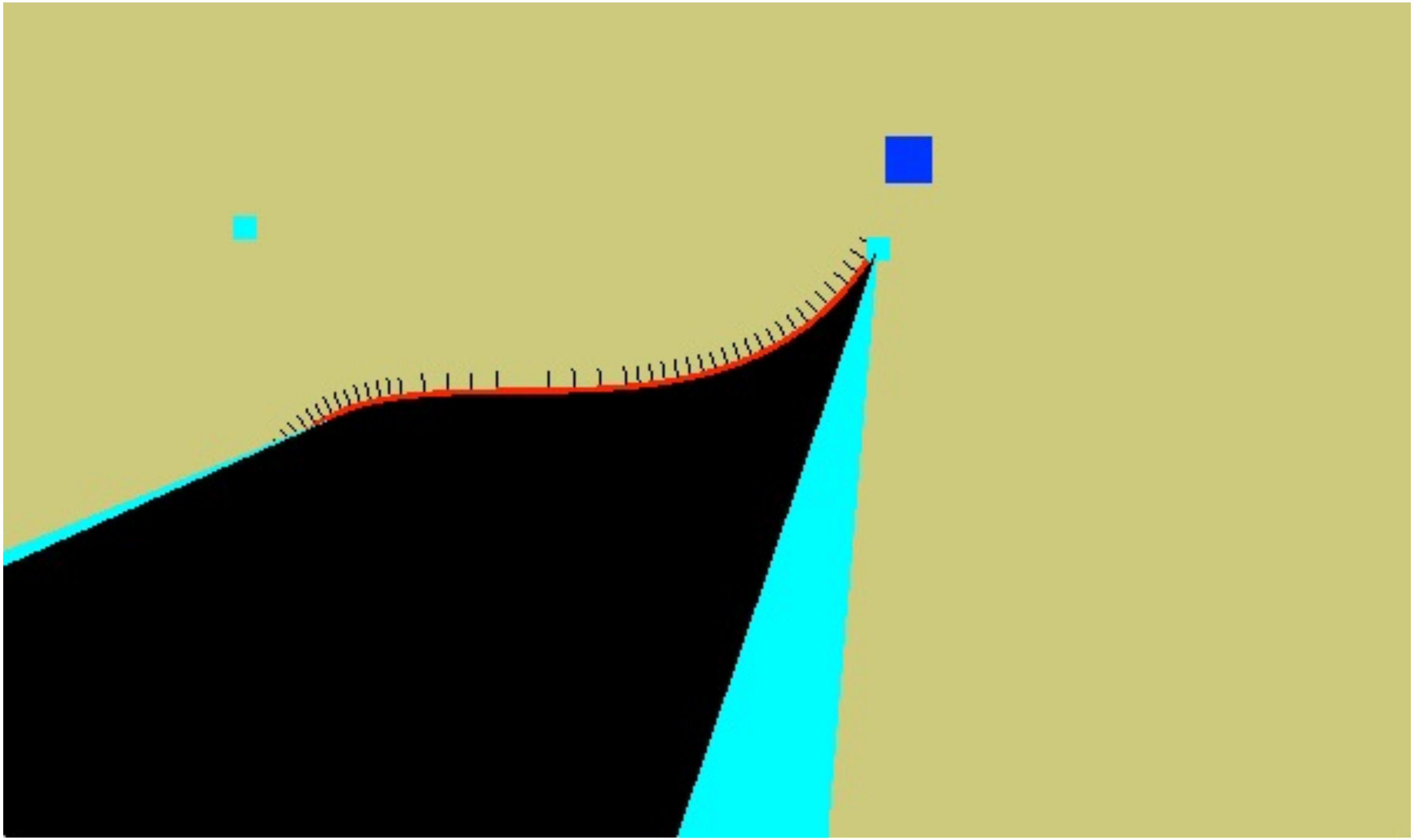


Case Study: Shadows and Lighting

- Recall gameplay prototype
  - Discrete shadows are easy
  - But had many problems
- Want something more robust
  - Continuously movement
  - Curved wall edges
  - Self-intersecting shadows
- Different features to test
  - Moving an avatar
  - Reconfiguring the wall

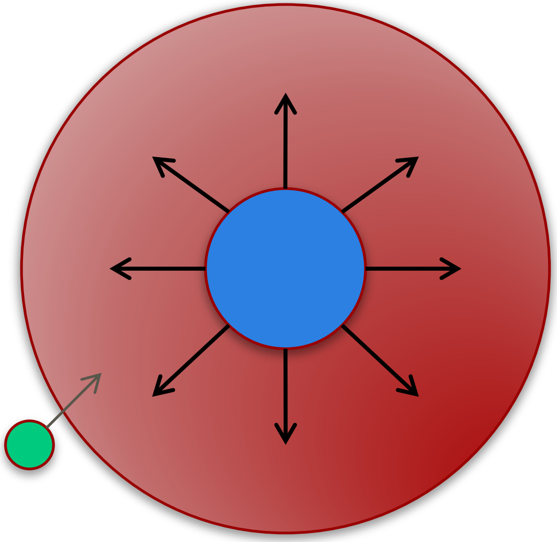


Case Study: Shadows and Lighting

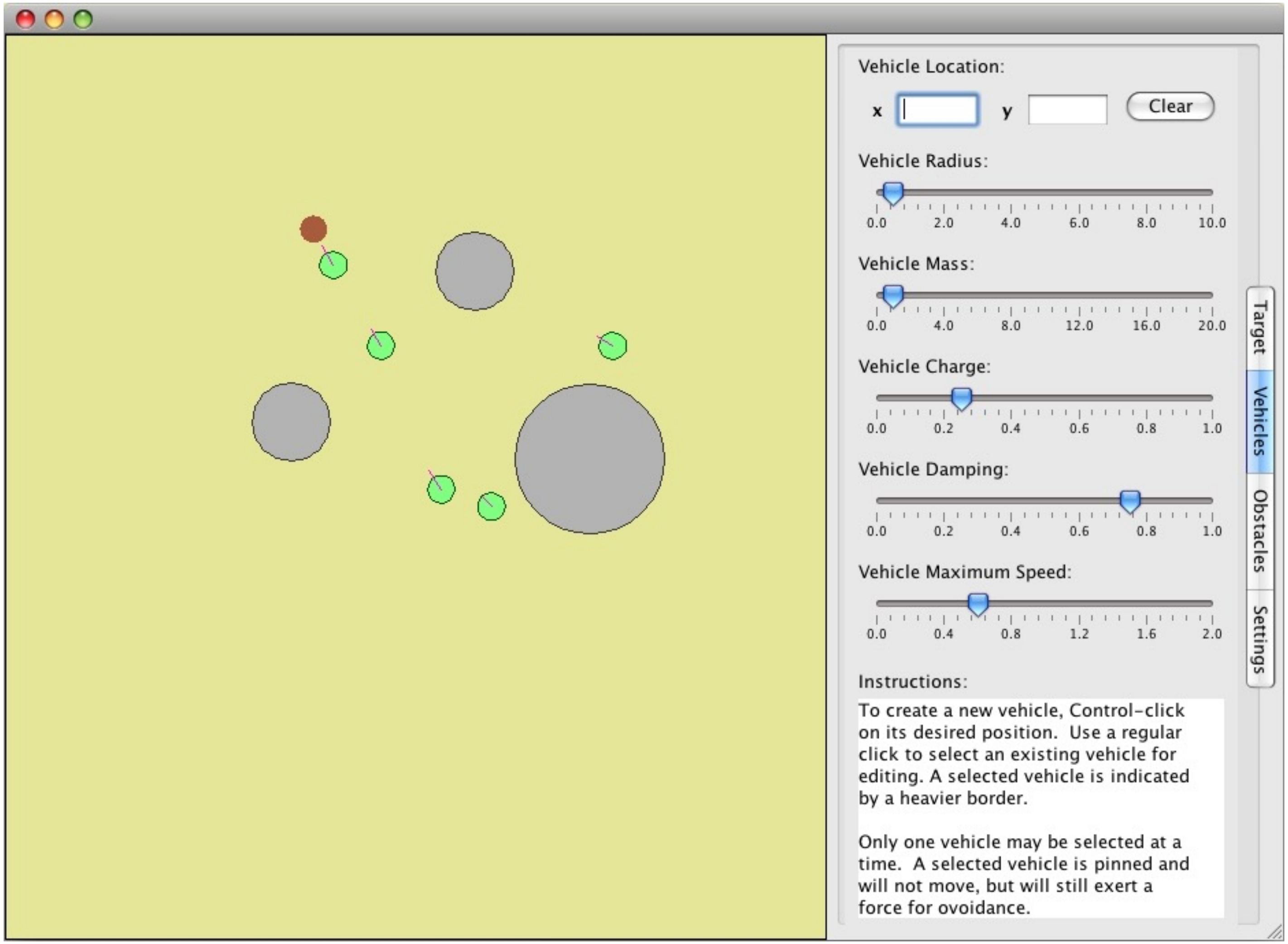


Case Study: Agent Movement

- Artificial potential fields
  - Obstacles are repulsive charge
  - Goal is an attractive charge
  - Sum together to get velocity
- Fast real-time movement
  - No hard AI algorithms
  - But has other problems…
- Will cover later in class
  - See *Pathfinding* in schedule



Case Study: Agent Movement



Vehicle Location:

x   y

Vehicle Radius:

Vehicle Mass:

Vehicle Charge:

Vehicle Damping:

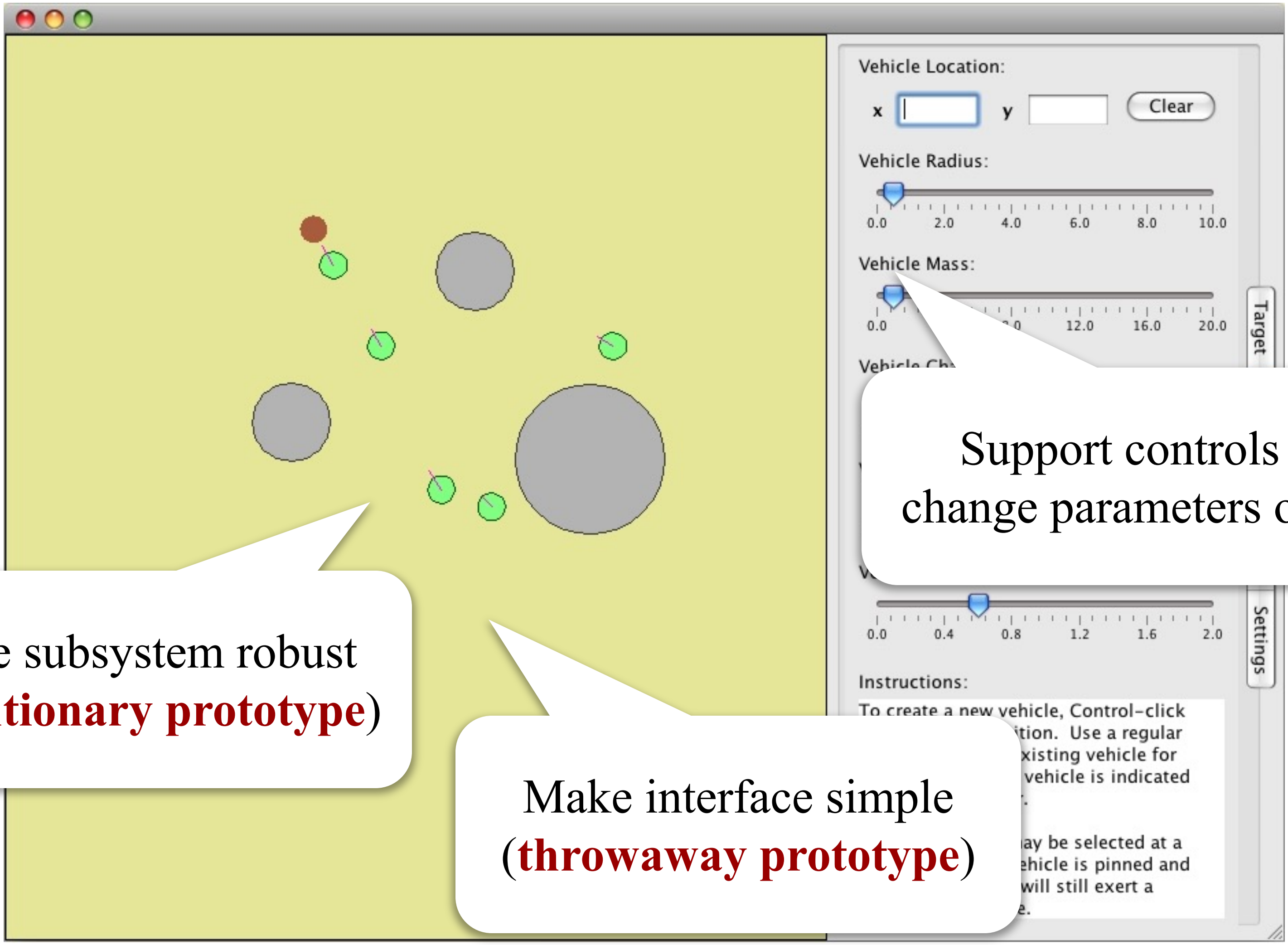
Vehicle Maximum Speed:

Instructions:

To create a new vehicle, Control-click on its desired position. Use a regular click to select an existing vehicle for editing. A selected vehicle is indicated by a heavier border.

Only one vehicle may be selected at a time. A selected vehicle is pinned and will not move, but will still exert a force for ovoidance.

Case Study: Agent Movement



To create a new vehicle, Control-click
tion. Use a regular
xisting vehicle for
vehicle is indicated

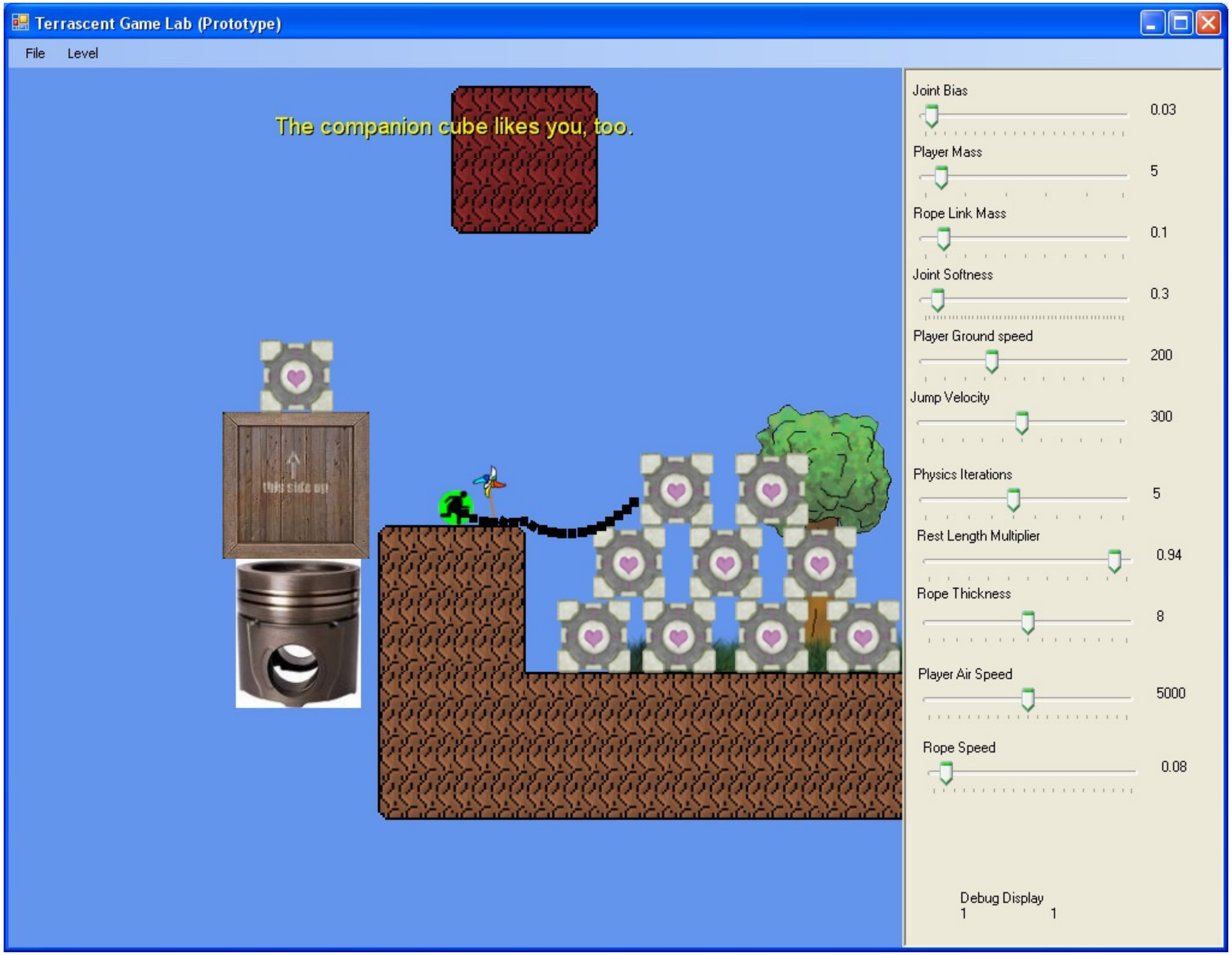
ay be selected at a
ehicle is pinned and
will still exert a
e.

Support controls to change parameters on fly

Make subsystem robust (**evolutionary prototype**)

Make interface simple (**throwaway prototype**)

3152 Example: *Forgotten Sky*



The companion cube likes you, too.

Joint Bias 0.03

Player Mass 5

Rope Link Mass 0.1

Joint Softness 0.3

Player Ground speed 200

Jump Velocity 300

Physics Iterations 5

Rest Length Multiplier 0.94

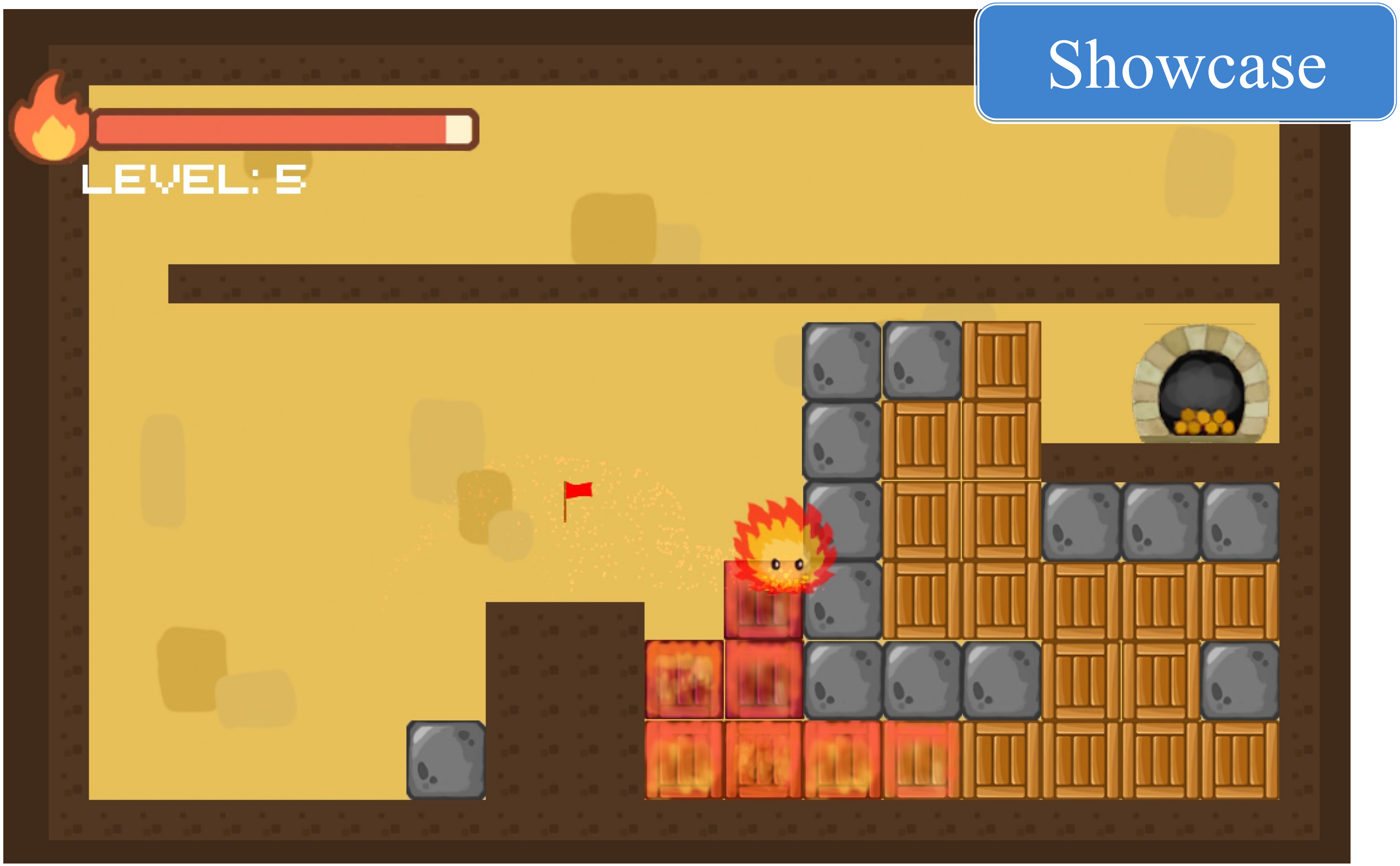
Rope Thickness 8

Player Air Speed 5000

Rope Speed 0.08

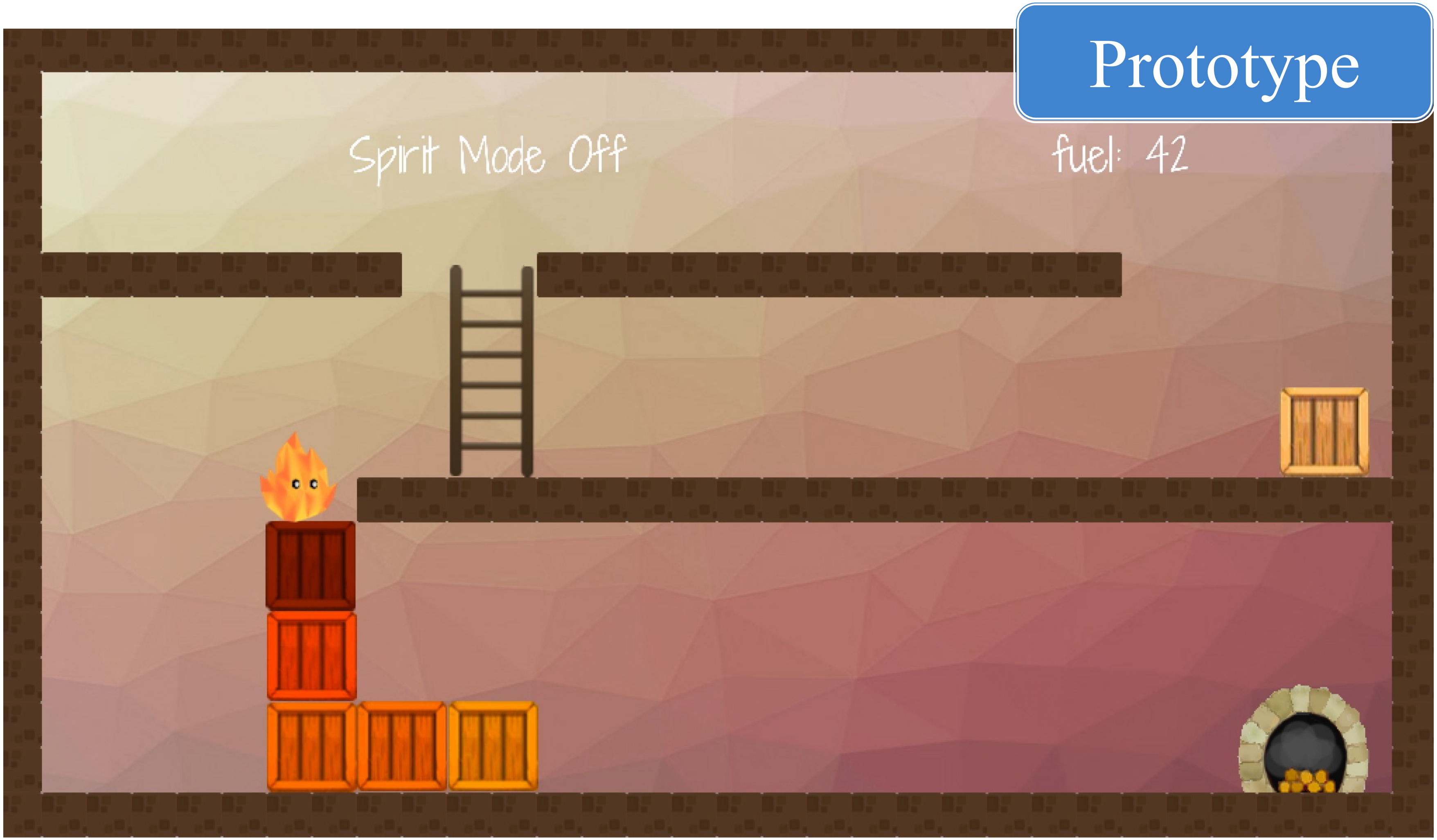
Debug Display
1 1

3152 Example: *Aiden*



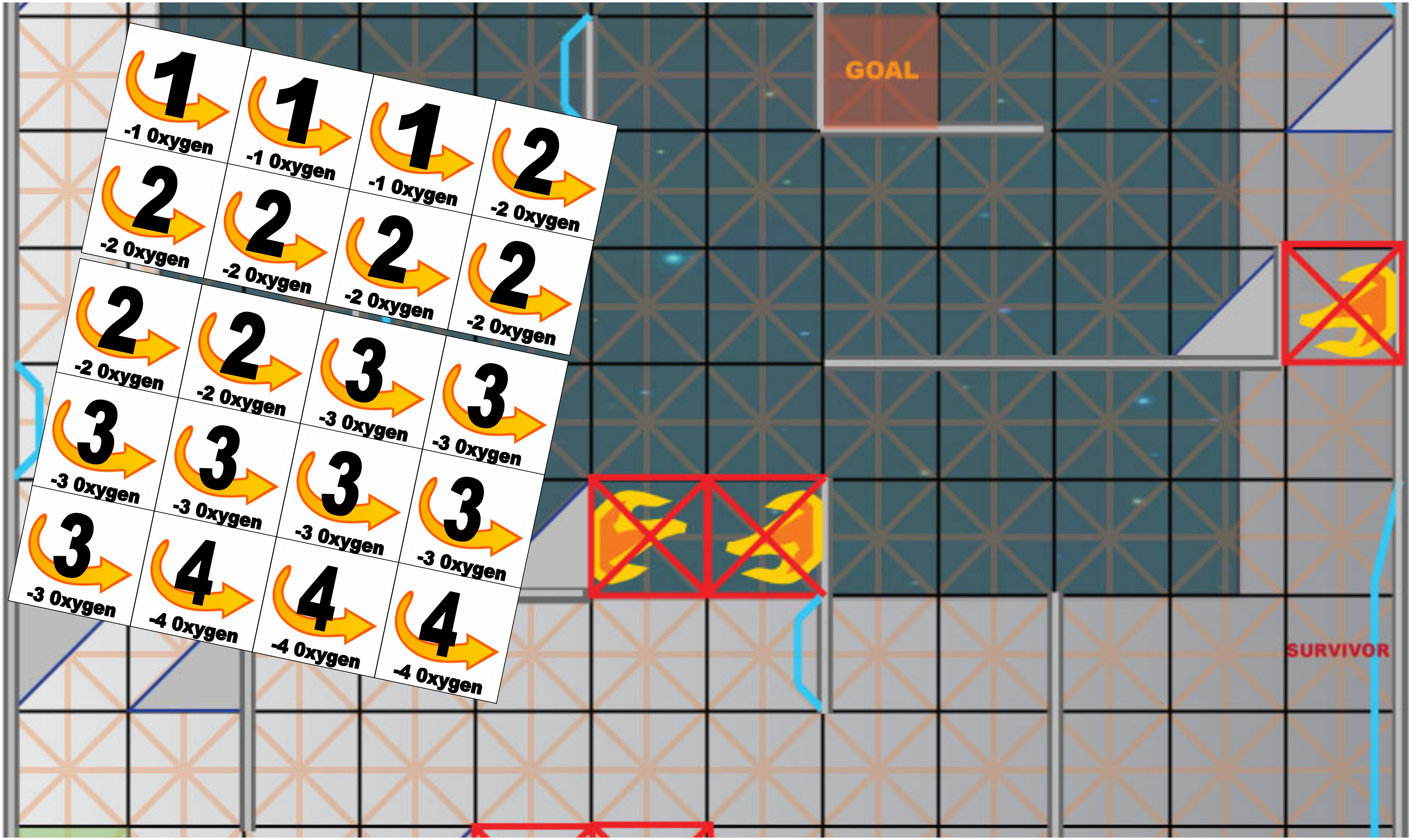
Showcase

3152 Example: *Aiden*



Prototype

Nondigital Prototypes



Digital or Nondigital?

Digital Prototypes

- Advantages
  - Closer to final design
  - Input and control semantics
  - Great for complex systems (e.g. physics)
- Disadvantages
  - Shuts out non-programmers
  - Longer development time

Nondigital Prototypes

- Advantages
  - Fast to create, iterate design
  - Used by non-programmers
  - Great for resources and game economy
- Disadvantages
  - Input and player control
  - Complex systems

Lessons From Nondigital Prototypes

- **Evaluate emergent behavior**
  - Allow player to commit simultaneous actions
  - Model interactions as “board elements”
- **Model player cost-benefit analyses**
  - Model all resources with sources and sinks
  - Focus on economic dilemma challenges
- **Early user testing for player difficulty**
  - Ideal for puzzle games (or puzzle element)
  - Can also evaluate unusual interfaces

Prototypes in this Class

- Required to demo three prototypes in class
  - **Nondigital prototype** week from Wednesday
  - **Gameplay prototype** on March 2nd
  - **Technical prototype** on March 17th
- Nondigital prototype may be trickiest
  - Keep it simple; avoid a full game
  - Focus on dilemma challenges (e.g. choice)
  - More details in the next lecture

The Gameplay Prototype

- **Throw-away prototype**
  - Does not have to be in Java
  - Can use another language (e.g. C#)
  - Can use authoring tools (e.g. HTML5, Unity)
- **Goal**: demonstrate gameplay
  - Challenges impossible in nondigital prototype
  - Basic player controls and interface
  - Primary game mechanic

The Technical Prototype

- **Evolutionary prototype**
  - Should be written in Java and LibGDX
  - Most of the code will be reused later
  - Some of code (e.g. interface) can be thrown away
- **Goal**: visualization and tuning
  - Simple interface displaying core functionality
  - Controls (e.g. sliders,console) to change parameters
  - Playtest to figure proper setting of parameters