OBJECT-ORIENTED THINKING

Topics

Topics

- The Object-Oriented Metaphor

Topics

- The Object-Oriented Metaphor
- Object-Oriented Flocks of Birds

Topics

- The Object-Oriented Metaphor
- Object-Oriented Flocks of Birds
  - Boids by Craig W. Reynolds

Topics

- The Object-Oriented Metaphor
- Object-Oriented Flocks of Birds
  - Boids by Craig W. Reynolds
- Modularity

The Object-Oriented Metaphor

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers
    - How do hundreds of individuals move in such coordinated behavior?

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position
    - Make global decisions about the behavior of the flock

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position
    - Make global decisions about the behavior of the flock
    - Figure out how those global decisions would affect each bird

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position
    - Make global decisions about the behavior of the flock
    - Figure out how those global decisions would affect each bird
  - **But in a real flock, each bird looks at her own environment, goals, etc. and makes decisions**

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position
    - Make global decisions about the behavior of the flock
    - Figure out how those global decisions would affect each bird
  - **But in a real flock, each bird looks at her own environment, goals, etc. and makes decisions**
    - Each bird acts as an individual based on her understanding of the world

The Object-Oriented Metaphor

- Consider Object-Oriented Programming as it relates to a flock of birds
  - **For hundreds of years, the motions of bird flocks (and schools of fish) fascinated researchers**
    - How do hundreds of individuals move in such coordinated behavior?
    - Which animal "leads" the flock? How is the leader chosen?
  - **To simulate a flock on a computer in a monolithic way, it was thought that a massive function would be required**
    - Track every bird's position
    - Make global decisions about the behavior of the flock
    - Figure out how those global decisions would affect each bird
  - **But in a real flock, each bird looks at her own environment, goals, etc. and makes decisions**
    - Each bird acts as an individual based on her understanding of the world
    - This is the core concept of Object-Oriented Programming (OOP)

Object-Oriented Flocks of Birds

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds
  - A simple, object-oriented approach to flocking behavior that Reynolds called "Boids"

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds
  - A simple, object-oriented approach to flocking behavior that Reynolds called "Boids"
  - Boids uses three simple rules:

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds
  - A simple, object-oriented approach to flocking behavior that Reynolds called "Boids"
  - Boids uses three simple rules:
    - Separation

Object-Oriented Flocks of Birds

- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds
  - A simple, object-oriented approach to flocking behavior that Reynolds called "Boids"
  - Boids uses three simple rules:
    - Separation
    - Alignment

Object-Oriented Flocks of Birds

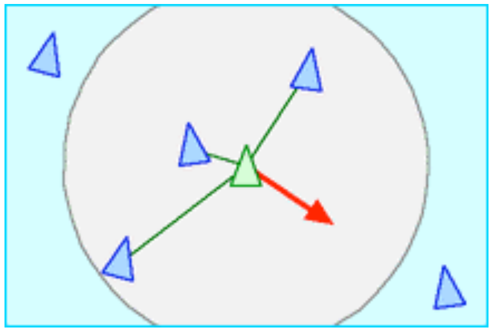
- BOIDS provides another approach to flocks
  - "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987) by Craig W. Reynolds
  - A simple, object-oriented approach to flocking behavior that Reynolds called "Boids"
  - Boids uses three simple rules:
    - Separation
    - Alignment
    - Cohesion

Object-Oriented Flocks of Birds

- Boids three rules

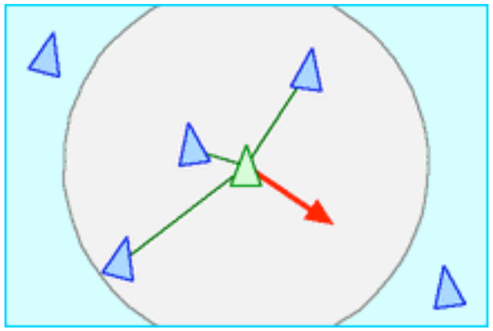
Object-Oriented Flocks of Birds

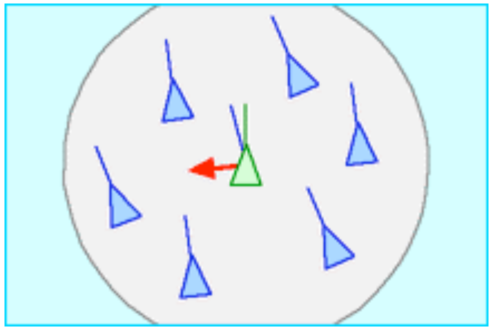
- **Boids three rules**
  - **Separation**
    - Avoid crowding nearby flockmates



Object-Oriented Flocks of Birds

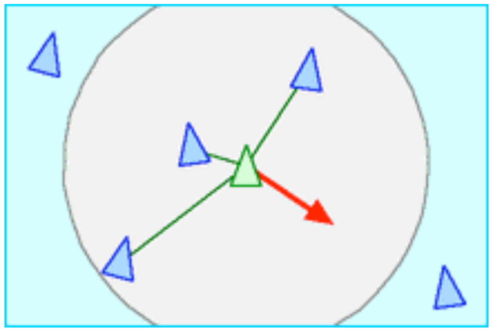
- **Boids three rules**
  - **Separation**
    - Avoid crowding nearby flockmates
  - **Alignment**
    - Match heading with nearby flockmates

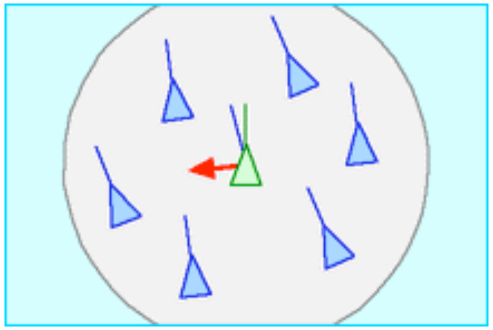


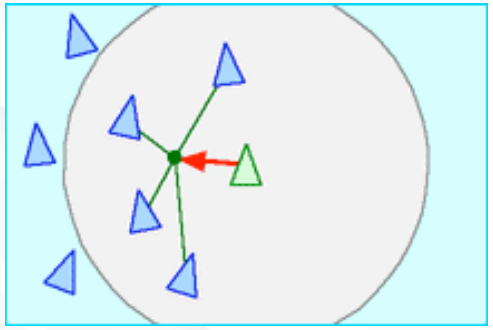


Object-Oriented Flocks of Birds

- **Boids three rules**
  - **Separation**
    - Avoid crowding nearby flockmates
  - **Alignment**
    - Match heading with nearby flockmates
  - **Cohesion**
    - Try to center self relative to nearby flockmates







Images are from Reynold's website http://www.red3d.com/cwr/boids/

Object-Oriented Flocks of Birds

PEARSON

Object-Oriented Flocks of Birds

- These three rules create surprisingly good flocking behavior

Object-Oriented Flocks of Birds

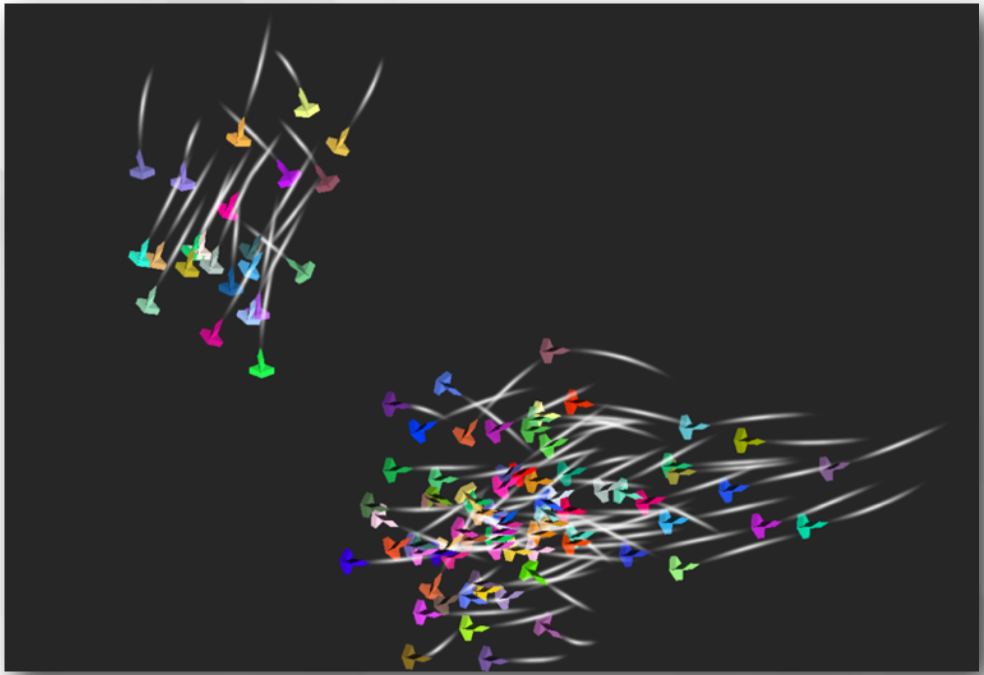
- These three rules create surprisingly good flocking behavior
  - https://www.youtube.com/watch?v=86iQiV3-3IA

Object-Oriented Flocks of Birds

- These three rules create surprisingly good flocking behavior
  - https://www.youtube.com/watch?v=86iQiV3-3IA
- In this chapter of the book, you create a simple 2D version of Boids that runs in Unity

Object-Oriented Flocks of Birds

- These three rules create surprisingly good flocking behavior
  - https://www.youtube.com/watch?v=86iQiV3-3IA
- In this chapter of the book, you create a simple 2D version of Boids that runs in Unity



Modularity

Modularity

- Another aspect of OOP is Modularity

Modularity

- Another aspect of OOP is Modularity
- Through modularity, each piece of code is expected to follow specific parameters

Modularity

- Another aspect of OOP is Modularity
- Through modularity, each piece of code is expected to follow specific parameters
  - But the internal implementation of that code is up to the individual developer

Modularity

- Another aspect of OOP is Modularity
- Through modularity, each piece of code is expected to follow specific parameters
  - But the internal implementation of that code is up to the individual developer
- This enables teams to effectively collaborate on code

Modularity

- Another aspect of OOP is Modularity
- Through modularity, each piece of code is expected to follow specific parameters
  - But the internal implementation of that code is up to the individual developer
- This enables teams to effectively collaborate on code
  - Each programmer receives a spec for what her part of the code is supposed to do

Modularity

- Another aspect of OOP is Modularity
- Through modularity, each piece of code is expected to follow specific parameters
  - But the internal implementation of that code is up to the individual developer
- This enables teams to effectively collaborate on code
  - Each programmer receives a spec for what her part of the code is supposed to do
  - That code is encapsulated into one or more classes which are clearly documented

Modularity

Modularity

- Collaboration benefits of modular code:

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class
  - The internals of the class can be replaced without affecting any other code

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class
  - The internals of the class can be replaced without affecting any other code
- However, top-down modularity is often difficult to define ahead of time in game prototypes

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class
  - The internals of the class can be replaced without affecting any other code
- However, top-down modularity is often difficult to define ahead of time in game prototypes
  - Because the spec for a prototype changes rapidly

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class
  - The internals of the class can be replaced without affecting any other code
- However, top-down modularity is often difficult to define ahead of time in game prototypes
  - Because the spec for a prototype changes rapidly
  - Instead of top-down, just think bottom-up about making your code easily reusable in later projects

Modularity

- Collaboration benefits of modular code:
  - The public fields and methods of each class are pre-specified
  - Other programmers can write code that will interact with each class without knowing the internals implementation of that class
  - The internals of the class can be replaced without affecting any other code
- However, top-down modularity is often difficult to define ahead of time in game prototypes
  - Because the spec for a prototype changes rapidly
  - Instead of top-down, just think bottom-up about making your code easily reusable in later projects
  - Make each reusable element of your code a module

Chapter 26 – Summary

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member
- From these simple rules can emerge complex, interesting behaviors

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member
- From these simple rules can emerge complex, interesting behaviors
- Modular code can help programming teams collaborate

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member
- From these simple rules can emerge complex, interesting behaviors
- Modular code can help programming teams collaborate

- Next Chapter: The Agile Mentality

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member
- From these simple rules can emerge complex, interesting behaviors
- Modular code can help programming teams collaborate

- Next Chapter: The Agile Mentality
  - Learn about how to approach small-team management

Chapter 26 – Summary

- In OOP, each instance of a class thinks for itself
  - Rather than all being controlled by a single function
- Through OOP, simple rules can be embedded into each class member
- From these simple rules can emerge complex, interesting behaviors
- Modular code can help programming teams collaborate

- Next Chapter: The Agile Mentality
  - Learn about how to approach small-team management
  - Learn to use burndown charts and scrum, which have had tremendous success on small, creative group projects