Scripting in Virtual Worlds with Remote Data

Behram Mistree

Virtual World Scripting

- Scripts bring objects to life
  - Cars
  - Houses
  - Clothing
  - Clocks
  - Etc.

Virtual World Scripting

- Scripts bring objects to life
- Simple example car.
  - Can create a car in a virtual world without scripting.
  - Scripting gets:
    - the car to turn on when you turn the key
    - accelerate when press the gas pedal
    - alarm owner when bing stolen

Virtual World Scripting

- Scripts bring objects to life
- Simple example car.
  - Can create a car in a virtual world without scripting.
  - Scripting gets:
    - the car to turn on when you turn the key
    - accelerate when press the gas pedal
    - alarm owner when bing stolen
- Changes world to dynamic, interactive space

Outline

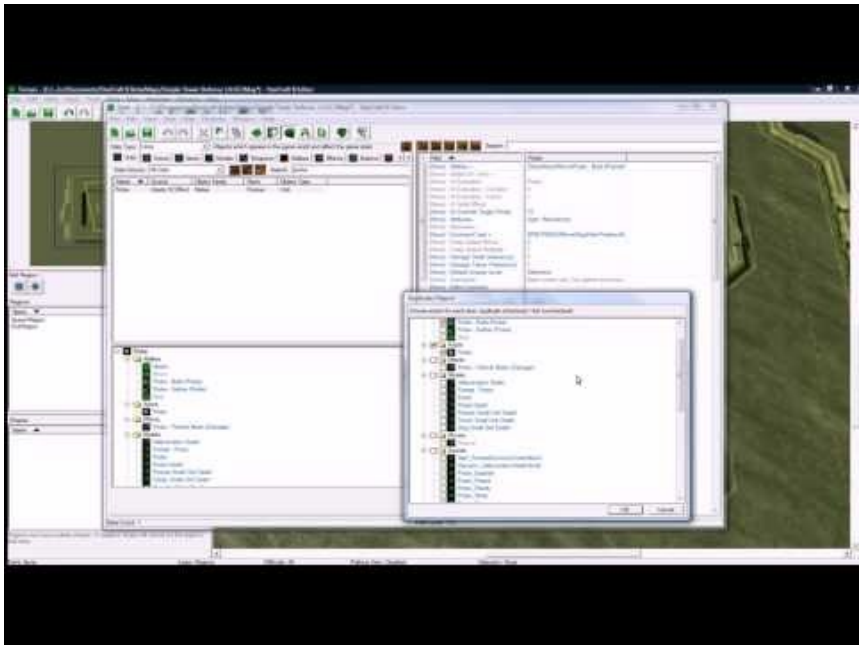
- Existing approaches
- Characteristics of a virtual world
- Data model
- Remote data
  - Definition
  - Approaches

Scripting: Existing Approaches

- Graphical front-ends
- General purpose languages
- (Occasionally) world-specific languages

Existing Approach: Graphical

- Examples: Starcraft editor, Madden player editor, etc.



Existing Approach: Graphical

- Greatly abstracts associated underlying language.
- Provide narrow customization
  - A character can only perform X-many attacks
  - One can only select certain types of characters
  - Cannot introduce other objects.

Existing Approach: Graphical

- Adequate for specific instantiations of worlds with well-defined narratives and purpose.
- Limits general expressiveness. Example in Starcraft:
  - Can't add a jukebox to world
  - It's possible to create creatures that will attack you or your opponents. It's unclear how a developer would create a new type of character that would only attack if attacked first.

Existing Approach: General Purpose Language

- Pros of this approach:
  - Developers may use pre-existing familiarity and pre-existing tools. (IDEs, tutorials, etc.)
  - Comparatively easier to deploy and port
  - Comparatively well-tested and well-supported interpreters and compilers.

Existing Approach: World-specific languages

- LindenScript
- Scalable Games Language
- Etc.

Outline

- ~~Existing approaches~~
- Characteristics of a virtual world
- Data model
- Remote data
  - Definition
  - Approaches

Target VW Qualities: Size

- World of Warcraft:

Target VW Qualities: Size

- World of Warcraft:
  - Over 10 million players on every continent but Antarctica

Target VW Qualities: Size

- World of Warcraft:
  - Over 10 million players on every continent but Antarctica
  - Over 10,000 servers

Target VW Qualities: Size

- World of Warcraft:
  - Over 10 million players on every continent but Antarctica
  - Over 10,000 servers
- State must be distributed

Target VW Qualities: User-developed content

- Basic examples:
  - Choosing characteristics of avatars (eg. Skills)
- Sophisticated examples:
  - Second Life scripts
  - World of Warcraft addons

Target VW Qualities: User-developed content

- Basic examples:
  - Choosing characteristics of avatars (eg. Skills)
- Sophisticated examples:
  - Second Life scripts
  - World of Warcraft addons

- Users have varied backgrounds, and cannot assume that, either through malice or ignorance, they add “incorrect” content.

Target VW Qualities: Approximate data

- Unlike a highly-detailed scientific simulation, the exact value may be approximate or stale.

Target VW Qualities: Approximate data

- Unlike a highly-detailed scientific simulation, the exact value may be approximate or stale.
- Examples:
  - Is object at <1,1,1> or <.999999,1,1>?
  - Did Event X occur 10 ms ago or 30 ms?

Target VW Qualities: Approximate data

- Unlike a highly-detailed scientific simulation, the exact value may be approximate or stale.
- Examples:
  - Is object at <1,1,1> or <.999999,1,1>?
  - Did Event X occur 10 ms ago or 30 ms?
- Avenue for managing complexity:
  - Doubles vs floats meaningless
  - Staleness can be tolerated if managed

Target VW Qualities: Continuous

- Action is always happening. Limited notion of:
  - Hard reboot/restart
  - Pause of execution

Target VW Qualities: Continuous

- Action is always happening. Limited notion of:
  - Hard reboot/restart
  - Pause of execution
  - Offline execution

Target VW Qualities: Continuous

- Action is always happening. Limited notion of:
  - Hard reboot/restart
  - Pause of execution
  - Offline execution

- Limits scope of potential applications
- Efficiency is important
- (With approximate data) Fast may be better than accurate

Target VW Qualities: Complexity

- Virtual worlds are asynchronous, and frequently many events happen apparently simultaneously.

  - Really important point. Language must be a tool for managing complexity: expose information, but not too much.
    - TMI: This turned a 10th of a degree. (May be important, may be unimportant depending on script writing. Must allow developers to write their own filters.)
    - There is a monster lurking behind a wall. (Different type.)

Target VW Qualities: Physics

- Virtual worlds all share some notion of physics.
  - In the most concrete way, we have a physical world with inescapable physical artifacts of motion, volume, etc.
  - Little consistency in either the engines used for physics, the guarantees they provide, or the physical laws the enforce.
    - Fly, walk, teleport, move through walls, not enter certain regions, etc.

Target virtual world characteristics

- Exceptions: There are existing virtual worlds that emphasize, de-emphasize, or do not support above bullets.
  - For example, cannot have user-generated content in EVE Online, lots aren't large, etc.

Outline

- ~~Existing approaches~~
- ~~Characteristics of a virtual world~~
- Data model
- Remote data
  - Definition
  - Approaches

Data Model: Architecture

- Space: Routes messages between objects on object hosts. Space analogies: SecondLife, EVE Online galaxy, etc.

Data Model: Architecture

- Space: Routes messages between objects on object hosts. Space analogies: SecondLife, EVE Online galaxy, etc.
- Objects: Have scripts that are executed on object hosts. Examples of objects: space ship, ice cream vendor, chess board.

Data Model: Architecture

- Space: Routes messages between objects on object hosts. Space analogies: SecondLife, EVE Online galaxy, etc.
- Objects: Have scripts that are executed on object hosts. Examples of objects: space ship, ice cream vendor, chess board.
- Object Hosts: Execute object scripts. Connect to spaces. Federated: may be owned/administered by anyone.

Data Model

- Each piece of data is managed by only one authoritative party: multiple objects for instance cannot directly write to the same variable.

Data Model: Objects

- Most data are managed by a single object.

Data Model

- Most data are managed by a single object.
  - Bank manages accounts of other entities
  - Switch controls state it's in (on or off).

Data Model: Space

- There is a special-class of data that is managed by the space itself. Geometric-type data. This is for two reasons:
  - Physics easier to do locally. Try figuring out object collisions.
  - Trust geometric data.
  - Space can actually enforce rules: cannot walk through walls, access restricted areas, etc.

Data Model: Space

- There is a special-class of data that is managed by the space itself. Geometric-type data. This is for two reasons:
  - Physics easier to do locally. Try figuring out object collisions.
  - Trust geometric data.
  - Space can actually enforce rules: cannot walk through walls, access restricted areas, etc.
- (Also a third-class of data: large, infrequently changing, which lives in a cdn, and won't be discussed.)

Data Model Summary

- Each piece of data is read/written by one authority.
- The space and other objects possibly on distant servers and outside of your administrative control serve as authorities

Outline

- ~~Existing approaches~~
- ~~Characteristics of a virtual world~~
- ~~Data model~~
- Remote data
  - Definition
  - Approaches

Data Model: Remote Data Definition

- Data is “remote” to an object if that object is not authoritative for it.
- Examples:
  - Positions of other objects
  - Hitpoints of an avatar in a war
  - Balance of a bank account

Remote Data: Concrete example – follow an object

<Video>

Remote data: Motivation

- Objects interact through and because of remote data.
  - An avatar buys a painting with his/her bank card
  - A wizard casts a healing spell on a soldier with low hit points

Remote Data: Problem

With data model, currently have isolated islands of objects and data.

- How do we pass information?
- When do we pass information?
- What information do we pass?
- Who initiates information passing?
- Who passes information? (Multicast throughout object hosts?)

Remote Data: Problem

With data model, currently have isolated islands of objects and data.

- How do we pass information?
- When do we pass information?
- What information do we pass?
- Who initiates information passing?
- Who passes information? (Multicast throughout object hosts?)

Lots of this: question of staleness

Remote Data: Two Semantic Uses

- On remote change, do something:
  - Example: when bank account drops below $100, add money to it.
- In course of action:
  - Example: before purchasing painting, check that bank account has sufficient funds.

Remote data: Goals

- System Perspective:
  - Small bandwidth, low computation, recycle closed connections/dynamically allocate, low control messages.
- Developer Perspective:
  - Automatic.
  - Transparent. (Glass box.)
  - Flexible.
  - Interoperable (do not necessarily have all object scripts on same type of hardware): may want to use message-passing to wrap whatever messages we're using. Allow other implementations (eg through message passing) to use same code interface.

Language construct vs. library

- Current language supports message passing.
  - Isn't that enough?
  - What is the value added for a new language construct?

Language construct vs. library

- Current language supports message passing.
- What is the value added for a new language construct?
  - Cleaner syntax
  - System-based optimization

Important: Can always use message-passing

- Can always use message-passing:
  - to meet more stringent requirements or
  - if have application-level knowledge of importance of updating variables.

- Language-level solution should appeal to quick “approximate” scripts.

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer
- Checkpoint state, and playback events according to timestamps of variables.

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer
- Checkpoint state, and playback events according to timestamps of variables
- Issue synchronous call

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer
- Checkpoint state, and playback events according to timestamps of variables.
- Issue synchronous call.
- Predict value.

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer
- Checkpoint state, and playback events according to timestamps of variables.
- Issue synchronous call.
- Predict value.
- Rely on pre-fetch

General Approaches to staleness

- Reduce effects by bounding error/constraining programmer
- Checkpoint state, and playback events according to timestamps of variables.
- Issue synchronous call.
- Predict value.
- Rely on pre-fetch.
- Make synchronous feasible

Reduce effects: Constrain programmer

- Why it's hard:
  - Programs are highly non-linear: a simple if-then-else block can produce abrupt changes.
- There are some models where this might work:

Reduce effects: Constrain programmer

- Why it's hard:
  - Programs are highly non-linear: a simple if-then-else block can produce abrupt changes.
- There are some models where this might work.
  - Every object phenotype is mapped to a linearly changing variable.

Checkpoint and playback

- Strategy: When receive a time-stamped update, go back to your state at that time, invalidate previous events, and replay with updated value.

Checkpoint and playback

- Strategy: When receive a time-stamped update, go back to your state at that time, invalidate previous events, and replay with updated value.
- Could be “correct”
- Huge overhead to save.
- For complex inter-dependencies, likely would spend most of your time invalidating data.

Predict Value

- Places in VW that already do this: predicting position of objects for instance.

Predict Value

- Places in VW that already do this: predicting position of objects for instance.
- Pre-supposes models for data or that data can be easily modeled.
  - Object at position <0,0,0> will be at <1,0,0> a second from now if it's traveling with velocity <1,0,0>.

Predict Value

- Places in VW that already do this: predicting position of objects for instance.
- Pre-supposes models for data or that data can be easily modeled.
  - Object at position <0,0,0> will be at <1,0,0> a second from now if it's traveling with velocity <1,0,0>.
  - Balance of a bank account? The health of an avatar? Etc.

Predict Value

- Requires some additional knowledge about data.
- Informs an optimization: timer that resets data value every minute. Can send that information to other object.
- Informs optimization 2: If you will generate an event that you know will change the value of the variable. For instance, sloppy implementation of TCP knows that if you send a FIN packet, state diagram will transition.

Predict Value

- Requires some additional knowledge about data.

Predict Value

- Requires some additional knowledge about data.

- Optimization: Timer that resets data value every minute. Can send that information to other object.

Predict Value

- Requires some additional knowledge about data.

- Optimization: Timer that resets data value every minute. Can send that information to other object.
- Optimization 2: If you will generate an event that you know will change the value of another variable, update variable
  - Sloppy implementation of TCP knows that if you send a FIN packet, state diagram will transition.

Predict Value

- Requires some additional knowledge about data.

- Optimization: Timer that resets data value every minute. Can send that information to other object.
- Optimization 2: If you will generate an event that you know will change the value of another variable, update variable
  - Sloppy implementation of TCP knows that if you send a FIN packet, state diagram will transition.
- Optimization 3: Provide semantics for quality of service **types** for data. Could be analogized to leases.

Pre-fetch Coordinator

- Create a function call graph with remote variable dependencies.

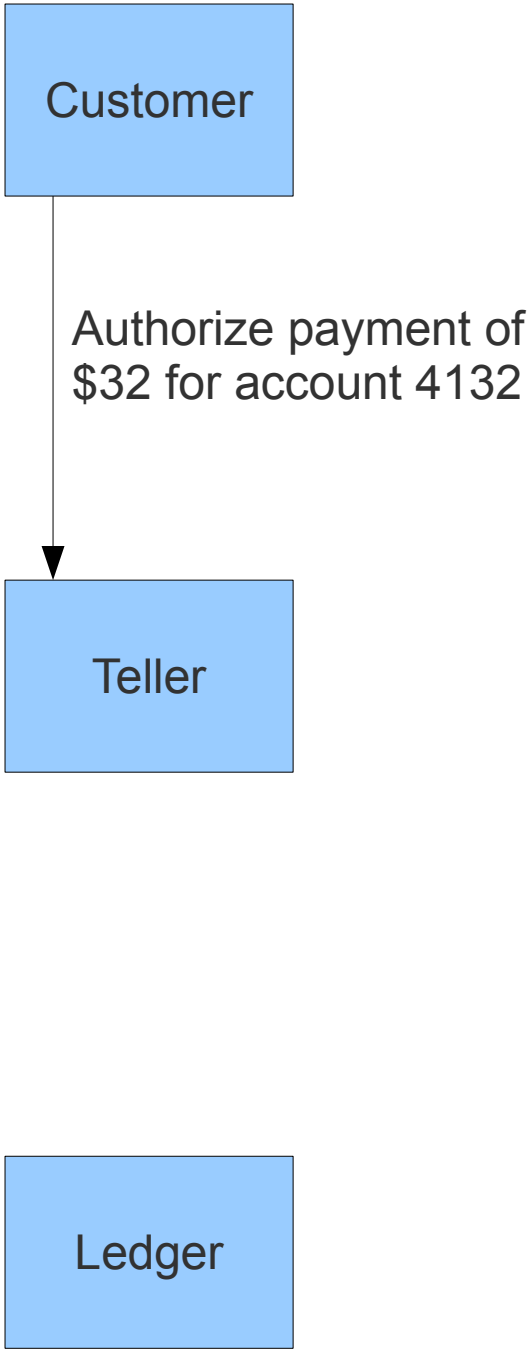
- Based on what events have been triggered, coordinator knows what data is needed, and updates object hosts.

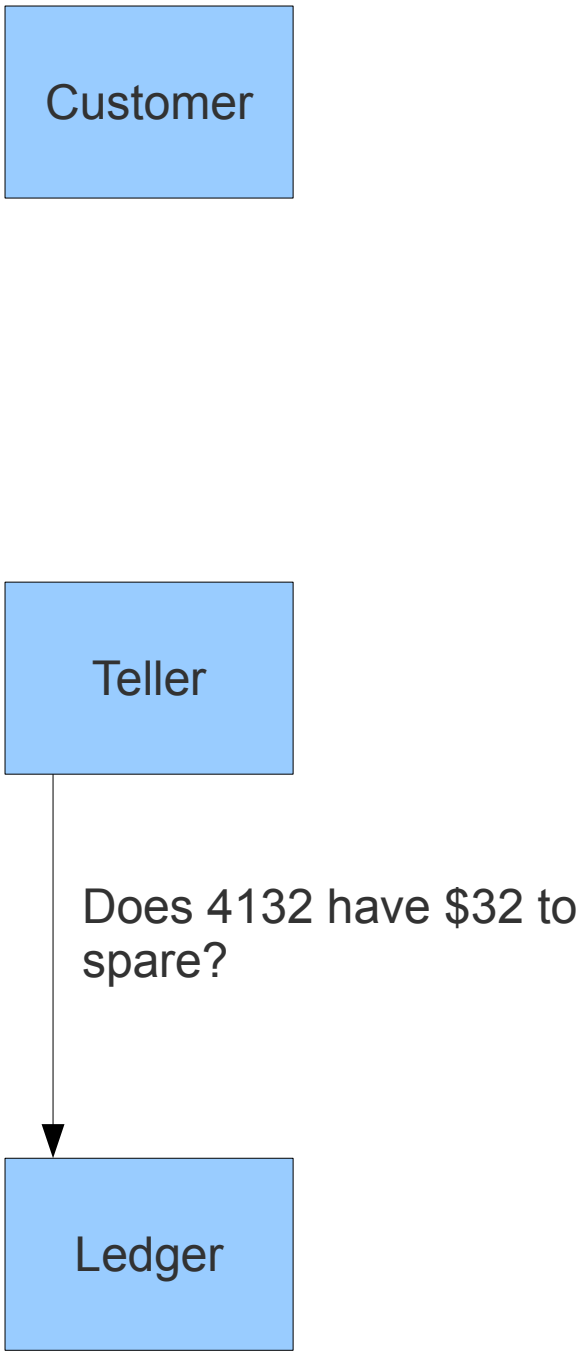
Pre-fetch Coordinator Example: Bank

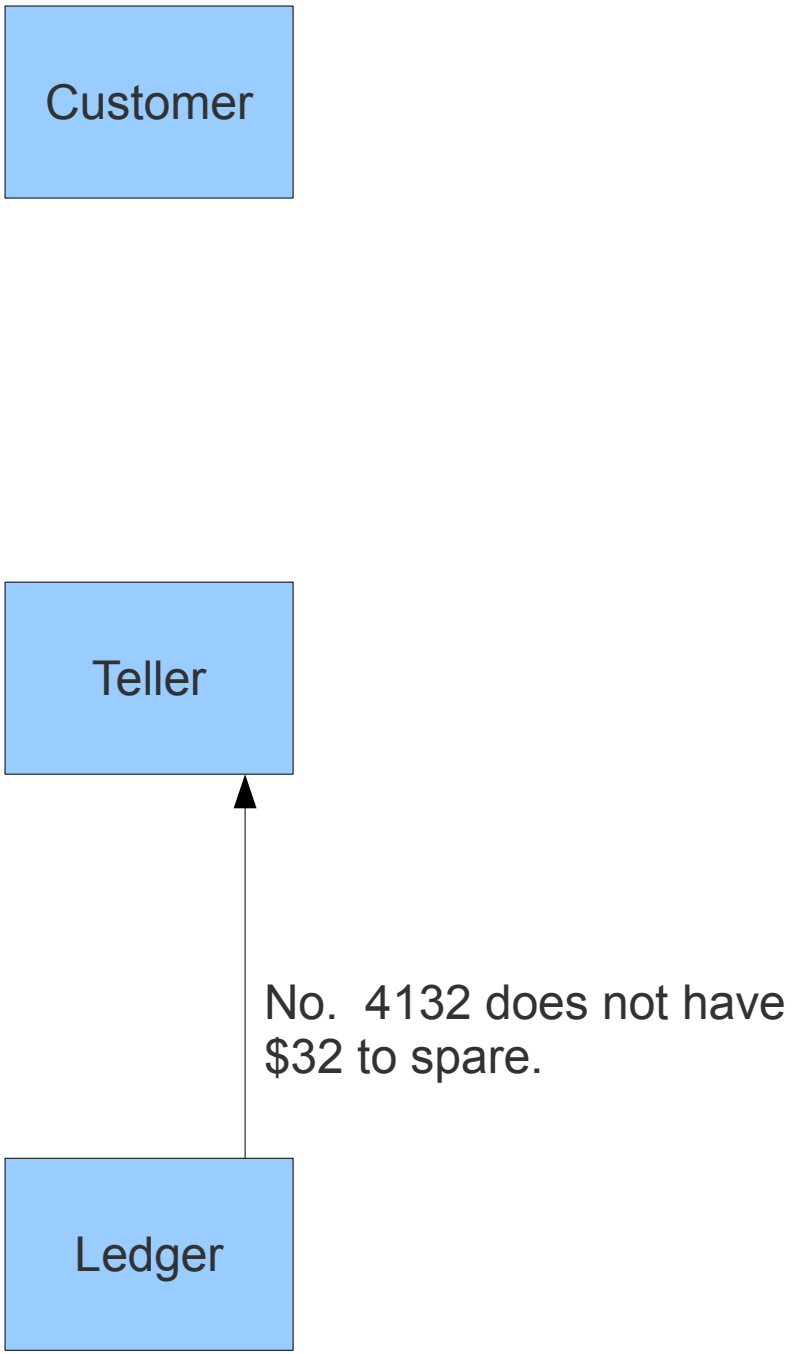
- Objects:
  - Customer – Uses account to pay for items
  - Teller – Signs off on purchases
  - Ledger – Precise record of customer balance

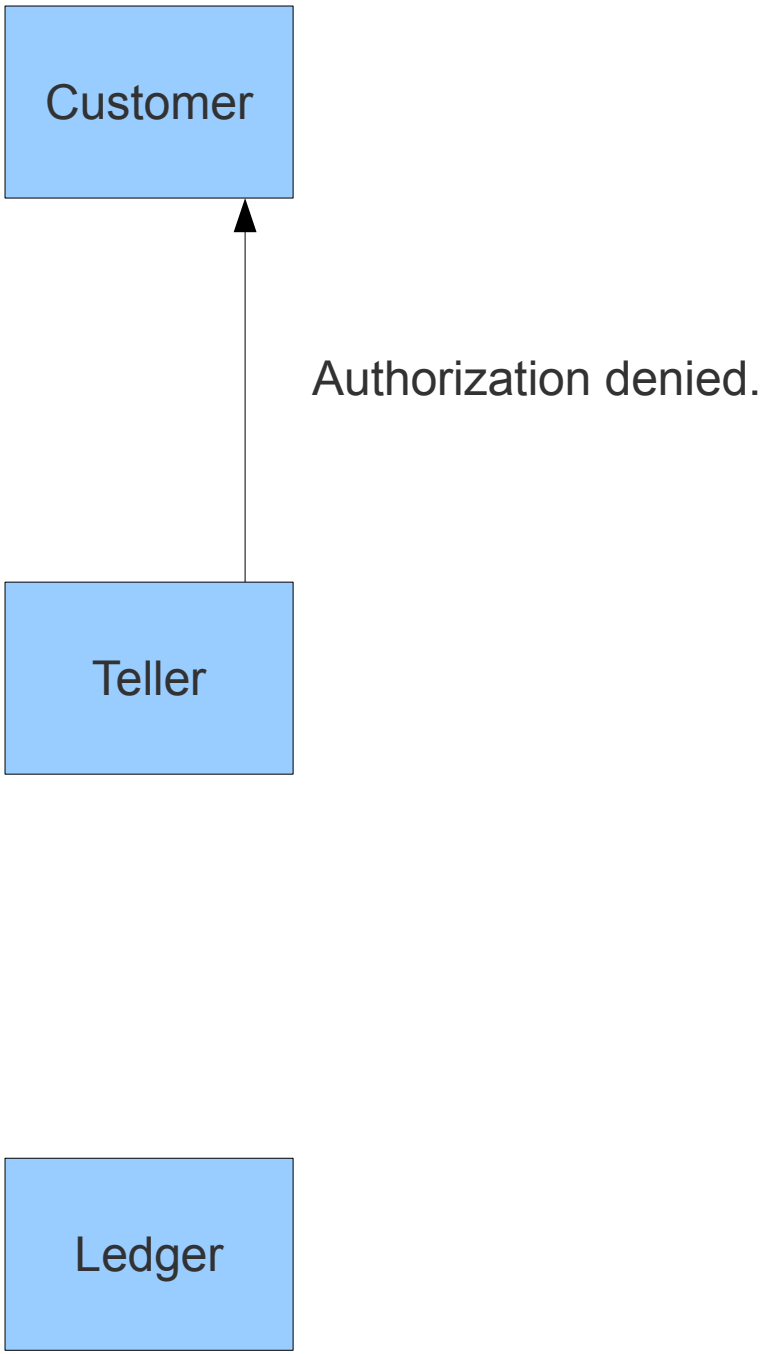
- Scenario
  - Customer calls buy function.

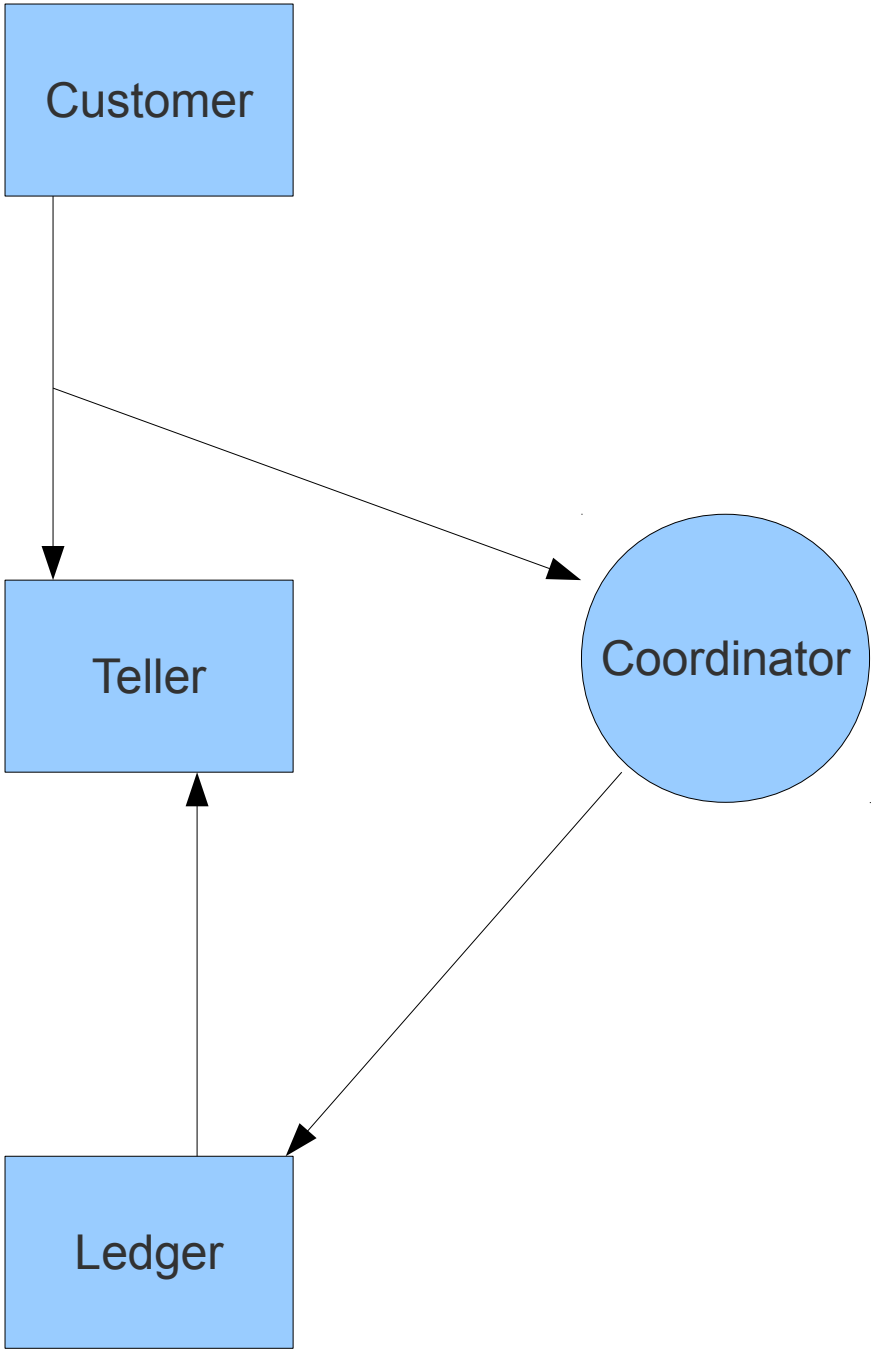

Customer
Teller
Ledger


Customer
Authorize payment of
$32 for account 4132
Teller
Ledger


Customer
Teller
Does 4132 have $32 to spare?
Ledger


Customer
Teller
No. 4132 does not have
$32 to spare.
Ledger


Customer
Authorization denied.
Teller
Ledger


Customer
Teller
Coordinator
Ledger

Pre-fetch Coordinator

- Create a function call graph with remote variable dependencies.
- Based on what events have been triggered, coordinator knows what data is needed, and updates object hosts.
- Distributed JIT

Pre-fetch Coordinator

- Create a function call graph with remote variable dependencies.
- Based on what events have been triggered, coordinator knows what data is needed, and updates object hosts.
- Distributed JIT
- Note: coordinator would not need to be global. Could be per object host.

Issue Synchronous Call I: Over network

Suggestion for optimizations: certainly wouldn't be a problem if were on same machine. There might be a little overhead, but tiny compared to network delay.

Issue Synchronous Call II: Other

- Create a scheme so that objects with inter-related remote data migrate to same server.

- Or segments of code are migrated over to run back and forth on alternate. (Optimization for session-like communication. Could not generally be relied upon.)

Going forward

- Use Emerson library for remote data.
- Begin building following optimizations:
  - If on same object host, can fetch the value directly.
  - Create types for leasing: distinguish between urgent updates and lazy updates.
  - Pre-fetch for timers
  - Examine workload from programmers this summer.