



unite'08

Techniques for Making Reusable Code

Mike Mittelman, Chief Architect, Hangout Industries



unite'08

Why reusable code?

- Lower cost: reusing code saves you time on the next project
- Fewer bugs: you tested it last time, didn't you?
- A team of one developer is just like a team of ten developers, but spread over time
 - “What did I have for breakfast yesterday?” becomes “What was I thinking when I wrote that?”
 - What does `RunThisOnceAndOnlyOnce()` do?

The logo for the 'unite '08' conference, featuring the text 'unite '08' in a white, sans-serif font. The text is partially overlaid by a stylized, abstract graphic of a bird or wing in shades of red and blue, with a white outline. The graphic is positioned behind the text, creating a layered effect.

unite '08

Reusable Code Strategies

- Decouple classes
- Remove public `GameObject` or `Component` references
- Compile-time errors are always better than run-time errors
- Use the event model



unit 8

Decoupling your code

- Smaller classes
- Smaller classes
- Interfaces
- Interfaces
- Private members



unit '08

Smaller classes

- The name of the class can mean something
- The bigger the class, the less reusable it is
- The less obvious the class, the less reusable it is

A graphic element consisting of several overlapping, curved, flame-like shapes in shades of red, orange, and blue, positioned behind the text 'Unit '08'.

Unit '08

Interfaces

- Describe the methods available to calling classes
- Instead of requiring a specific class, you require specific methods
- Allows greater flexibility for how methods are implemented
- Excellent decoupling technique

Remove public references

- Get rid of “public GameObject foo”
- Get rid of “public Component foo”
- Both of these are definitely tightly coupled code
- The hierarchy of the scene should be irrelevant to the code
 - That means transform.parent has to go too
 - Exception: if the hierarchy was created by code

The logo for the 'unite '08' conference, featuring the text 'unite '08' in a white, sans-serif font. The text is partially overlaid by a stylized, abstract graphic of a bird or wing in shades of red and blue, with a white outline. The background is black.

unite '08

Public references problems

- Very hard to debug
- Requires moving Game Objects into new scene or project
- Can be easily broken
- Run time errors (yuck!)



unit 8

The event model

- Used throughout ActionScript
- Events are an implied interface
- “Let me know if something happens”
- “Something happened, deal with it”
- Excellent for web-based projects

Events are “three state data”

- Properties and members are often “null” when not yet initialized
- “null” might mean other things
- Events are clearer in meaning
- Events do not require a coroutine
- Three states:
 - Unavailable
 - Available
 - Updated

The logo for 'unite'08, featuring the text 'unite'08 in a white, sans-serif font. The text is partially overlaid by a stylized graphic of a bird or flame in shades of red and blue, with a white outline. The graphic is positioned behind the text, creating a layered effect.

unite'08

Event example:

- Hangout.net UserChangeMessage
- List of all users in the room, and their properties
- Can change at any time
- Not necessarily available at startup
- Obtained externally

The logo for 'unite'08, featuring the text 'unite'08 in a white, sans-serif font. The text is partially overlaid by a stylized graphic of a bird or a flame in shades of red and blue, with a white outline. The graphic is positioned behind the text, creating a layered effect.

unite'08

Event rules:

- Never expose data you received as an event as a property
 - You can't convert three state data to simple data
- Be prepared for extra events
 - You might get the same news twice, but that shouldn't be an issue
- If nobody should have access to an instance before an event, expose it through the event

The logo for 'unite'08, featuring the text 'unite'08 in a white, sans-serif font. The text is partially overlaid by a stylized graphic of a bird or a flame in shades of red and blue, with the '0' being a large, stylized zero.

unite'08

Our event implementation

- Singleton class for registering listeners
- Many message classes
- Message is type and contains data
- Function to register listeners uses generic
- Compile-time errors (yay!), run-time silence (um, yay?)
- Remove listeners in onDisable
 - References to delegates will keep object from being disposed



unite '08

Code Example

```
private void Awake() {  
    Room.current.AddListener<UserChangeMessage>( GetLocalAvatar );  
}
```

```
private void Start() {  
    Room.current.Route( new UnityInitMessage() );  
}
```

```
private void OnDisable() {  
    Room.current.RemoveListener<UserChangeMessage>( GetLocalAvatar );  
}
```

```
private void GetLocalAvatar( UserChangeMessage inMessage ) {}
```

The logo for 'unite '08' features the text 'unite '08' in a white, sans-serif font. The text is partially overlaid by a stylized graphic of a bird or a flame in shades of red and blue, with sharp, pointed edges.

unite '08

Questions?

Mike Mittelman

Chief Architect

Hangout Industries

mike.mittelman@hangout.net

www.hangout.net