Knowledge Overload
Keeping AI Knowledge Organized In Large Scale Projects

Guillaume Bouilly
SQUARE ENIX CO., LTD.
2020/09/03

Speaker Introduction

- Academic background
  - Software Engineer
  - AI specialization (M. Sc. A)
- Eidos Montréal (2012 – 2018)
  - Deus Ex : Mankind Divided (2016)
- SQUARE ENIX CO., LTD (2018 – present)
  - Advanced Technology Division

Speaker Introduction

AI Systems I have worked on

- Patrol behaviors
- Perception
- Investigation behaviors
- Stimulus tracking and prioritization
- Threat management
- Search behaviors
- Bark system
- Companion AI
- Ranged combat
- Combat positioning
- Squad behaviors
- Navigation
- Aiming and look-at
- Ambient AI
- Scripted Events
- Conversation Systems

Speaker Introduction

- To me the most important part of game AI is

Knowledge

AI Knowledge

- A simple definition
  - AI Knowledge is data organized in a way that allows the AI to understand its environment.
  - This data can be used by AI systems to let the AI perform some actions.

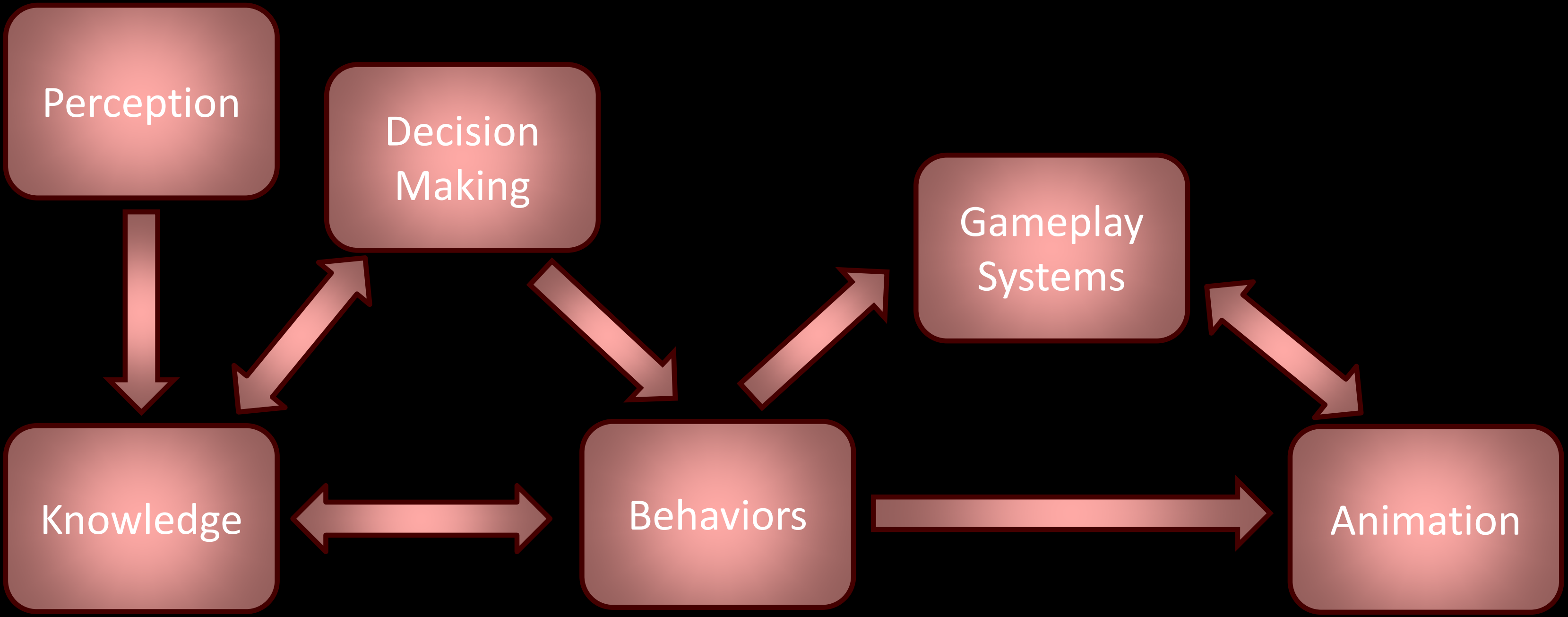
AI Knowledge

- Why is knowledge important?
  - Basis of decision making
  - Limitations are often linked to knowledge

AI Knowledge

- Knowledge Representation
  - Navigation Data
  - Influence Map
  - Reasoning Grid
  - Blackboard
  - Etc…

AI Knowledge



AI Knowledge

- A lot of bugs are actually knowledge bugs
  - Wrong information
  - Out of date information
  - Human errors

AI Knowledge

- Why is this important?
- AAA games in 2020
  - Hundreds of employees
  - Several years
  - Multiple locations

AI Knowledge

- Pitfalls of inadequate knowledge management
  – Longer onboarding
  – Confusion and errors
  – Inefficient communication
  – Bugs

  => Increased development time

Outline

- Case 1 : Too Much Information!
- Case 2 : External Reasoning
- Case 3 : Dealing With Archetypes
- Case 4 : Asynchronous Updates
- Putting it all together
- Follow-up
- Additional Tips

Case 1

Too Much Information

Case 1 : Too Much Information

CurrentState
PreviousState
Mood
CurrentPatrol
NextPatrolPoint
IsJumping
JumpStartPosition
JumpEndPosition
JumpType
CurrentBark
LastBarkTimeStamp

CurrentThreat
IsVisible
LastKnownPosition
LastDetectedTimeStamp
RestrictedAreaType
RestrictedAreaFaction
WarningType
WarningTimer
StimulusType
StimulusTimestamp
StimulusPosition

InvestigationType
InvestigationPosition
GroupInvestigation
InvestigationTimer
CurrentTarget
CurrentPosition
CurrentCover
DestinationCover
CoverType
CoverAimingType
IsCoverDestructible
CoverToCoverOption

Faction
PlayerRelationship
JobType
SpecialSkill
MainWeapon
BackupWeapon
BulletType
Grenade
AimTarget
IsTargetBlocked
IsAiming

Case 1 : Too Much Information

- Problem
  - Too much information

Case 1 : Too Much Information

- Problem
  - Initial learning barrier too high
  - Too much time spent searching
  - Confusion between similar named elements

Case 1 : Too Much Information

CurrentState
PreviousState
Mood
CurrentPatrol
NextPatrolPoint
IsJumping
JumpStartPosition
JumpEndPosition
JumpType
CurrentBark
LastBarkTimeStamp

CurrentThreat
IsVisible
LastKnownPosition
LastDetectedTimeStamp
RestrictedAreaType
RestrictedAreaFaction
WarningType
WarningTimer
StimulusType
StimulusTimestamp
StimulusPosition

InvestigationType
InvestigationPosition
GroupInvestigation
InvestigationTimer
CurrentTarget
CurrentPosition
CurrentCover
DestinationCover
CoverType
CoverAimingType
IsCoverDestructible
CoverToCoverOption

Faction
PlayerRelationship
JobType
SpecialSkill
MainWeapon
BackupWeapon
BulletType
Grenade
AimTarget
IsTargetBlocked
IsAiming

Case 1 : Too Much Information

ThreatData

IsVisible
LastKnownPosition
LastDetectedTimeStamp

StimulusData

StimulusType
Timestamp
Position
InvestigationPosition

TargetData

CurrentTarget
CurrentPosition

JumpData

IsJumping
StartPosition
EndPosition

AimingData

AimTarget
IsTargetBlocked
IsAiming

BarkData

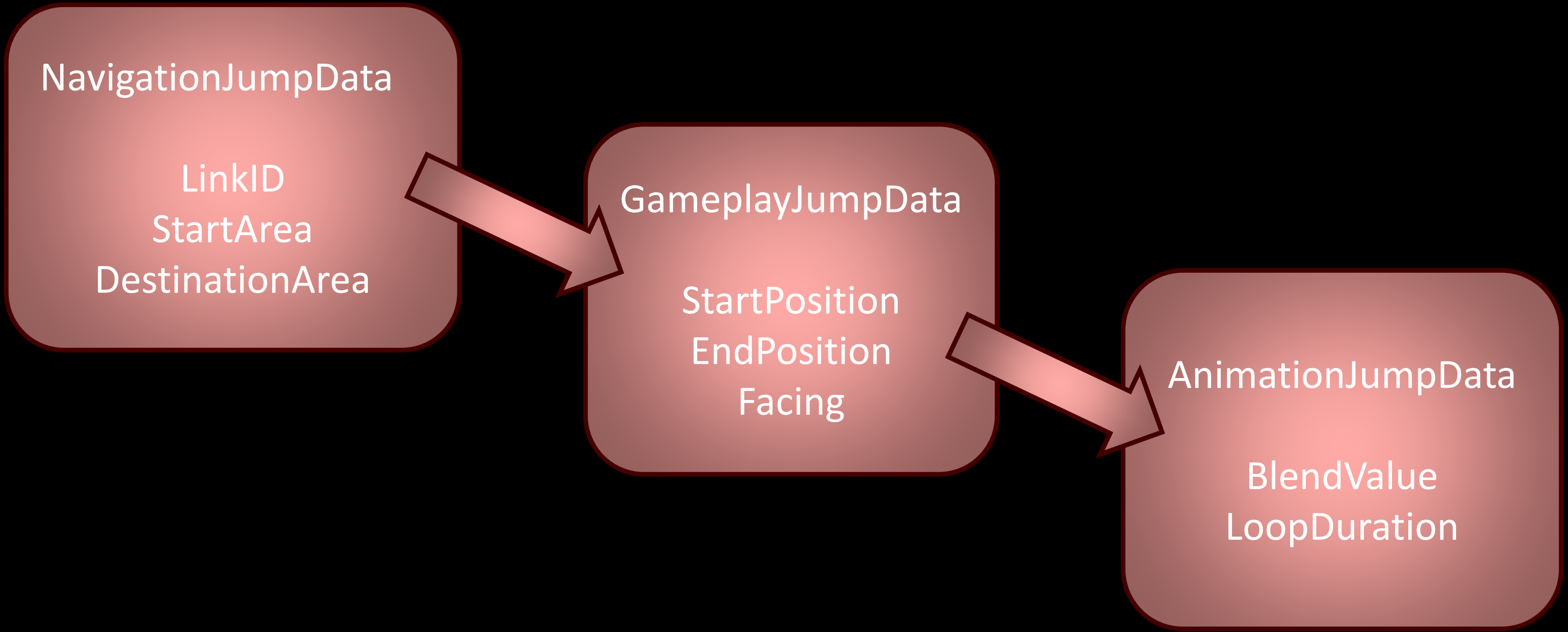
CurrentBark
LastBarkTimeStamp

Case 1 : Too Much Information

Jump Data

LinkID
StartArea
DestinationArea
StartPosition
EndPosition
Facing
BlendValue
LoopDuration

Case 1 : Too Much Information



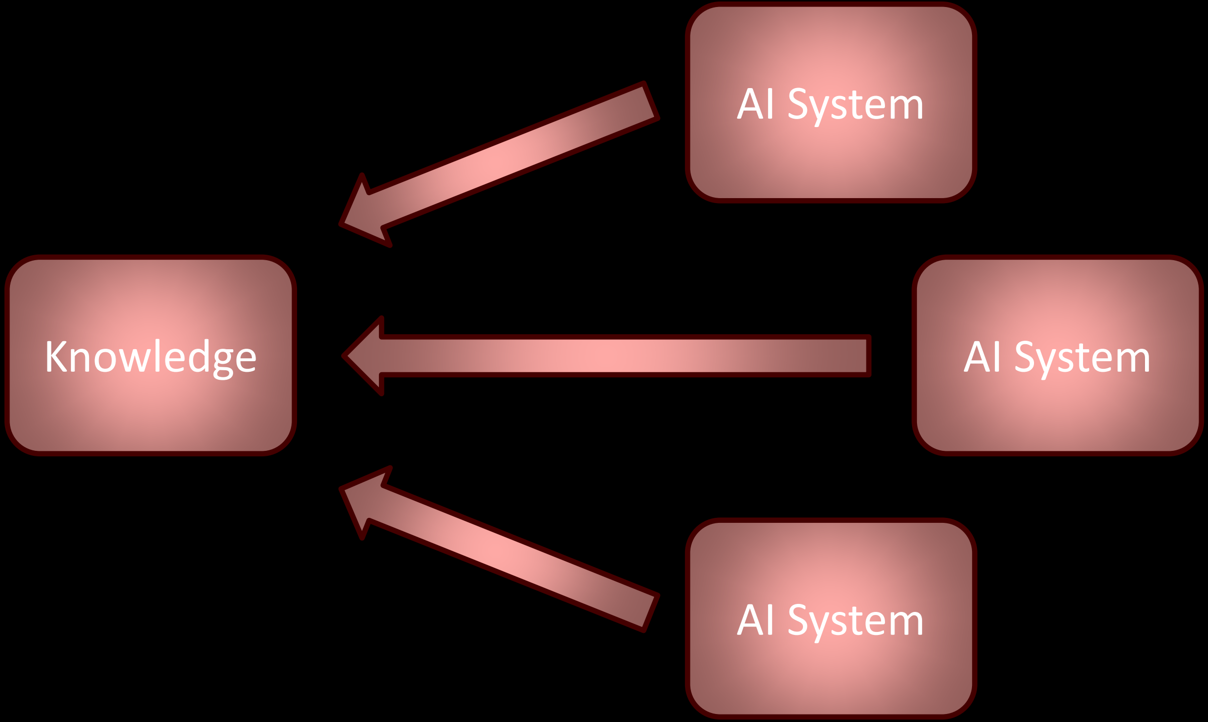
Case 1 : Too Much Information

- Lessons
  - Smaller pods of knowledge are easier to manage
  - Regroup what makes sense together

Case 2

External Reasoning

Case 2 : External Reasoning



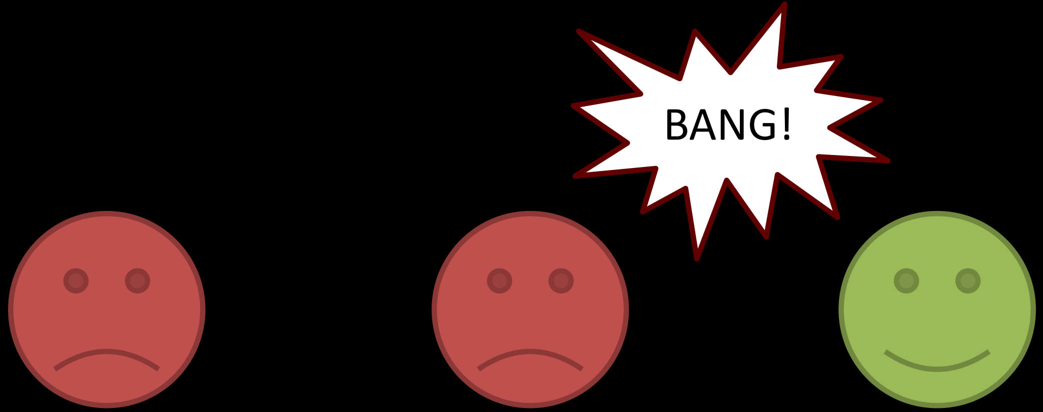
Case 2 : External Reasoning

Wandering

Using Objects

Conversation

Case 2 : External Reasoning

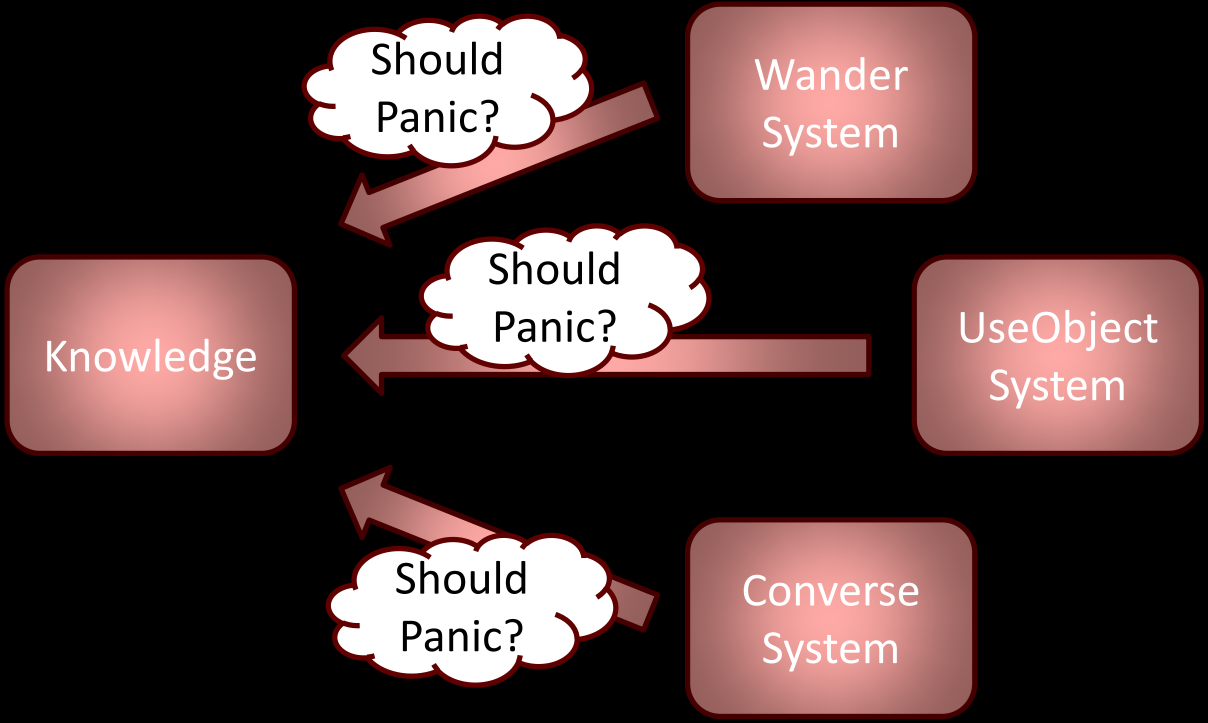


Flee

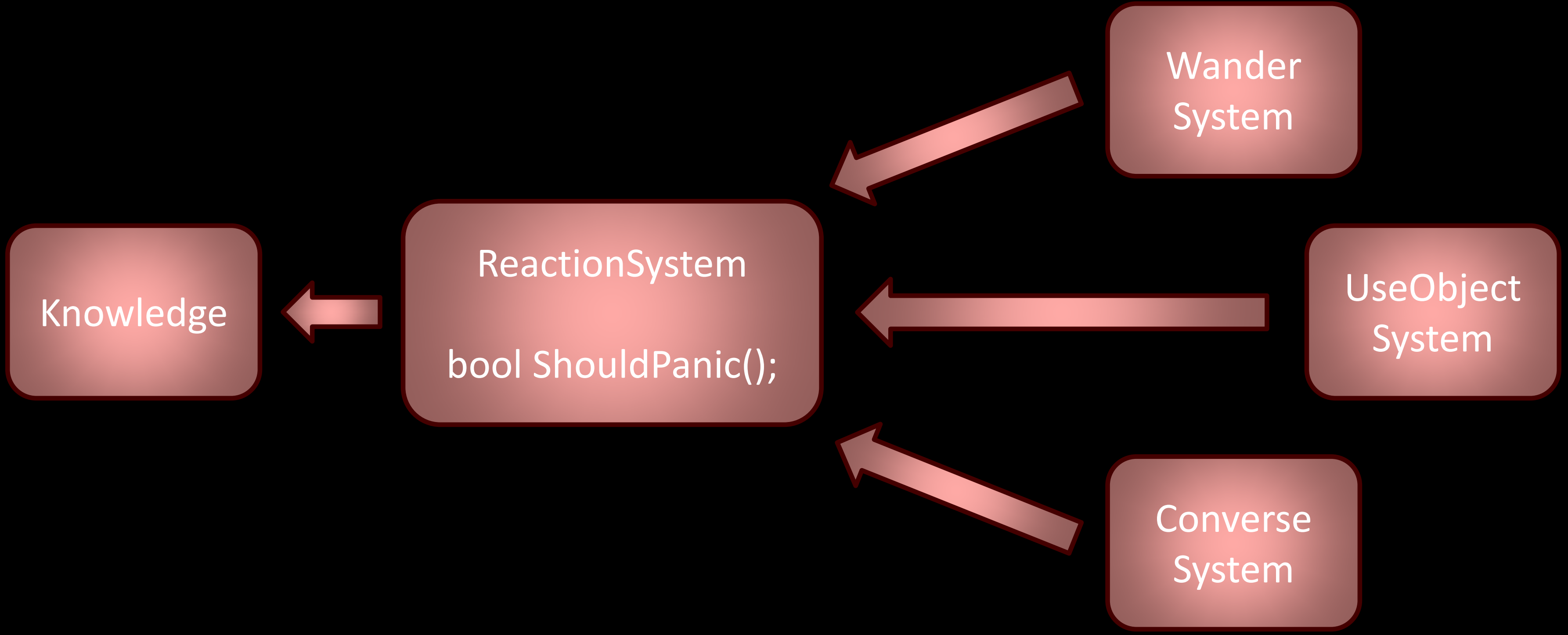
Flee

Conversation

Case 2 : External Reasoning



Case 2 : External Reasoning



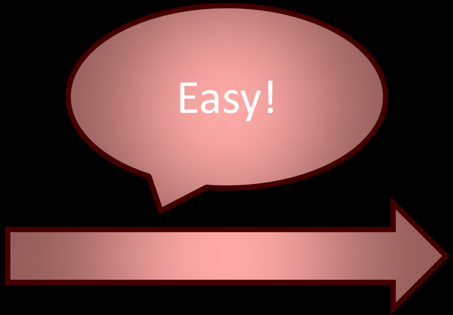
Case 2 : External Reasoning

- Lesson
  - Encapsulate the reasoning at only one location

Case 2 : External Reasoning

AI States

- Neutral
- Suspicious
- Alarmed
- Hostile



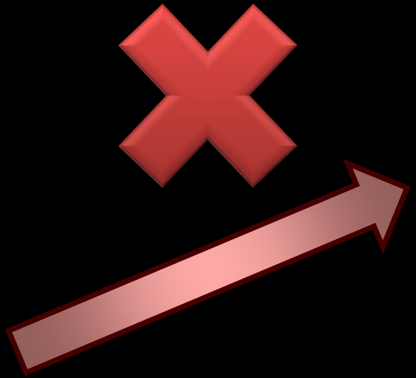
Radar

- Neutral
- Suspicious
- Alarmed
- Hostile

Case 2 : External Reasoning

AI States

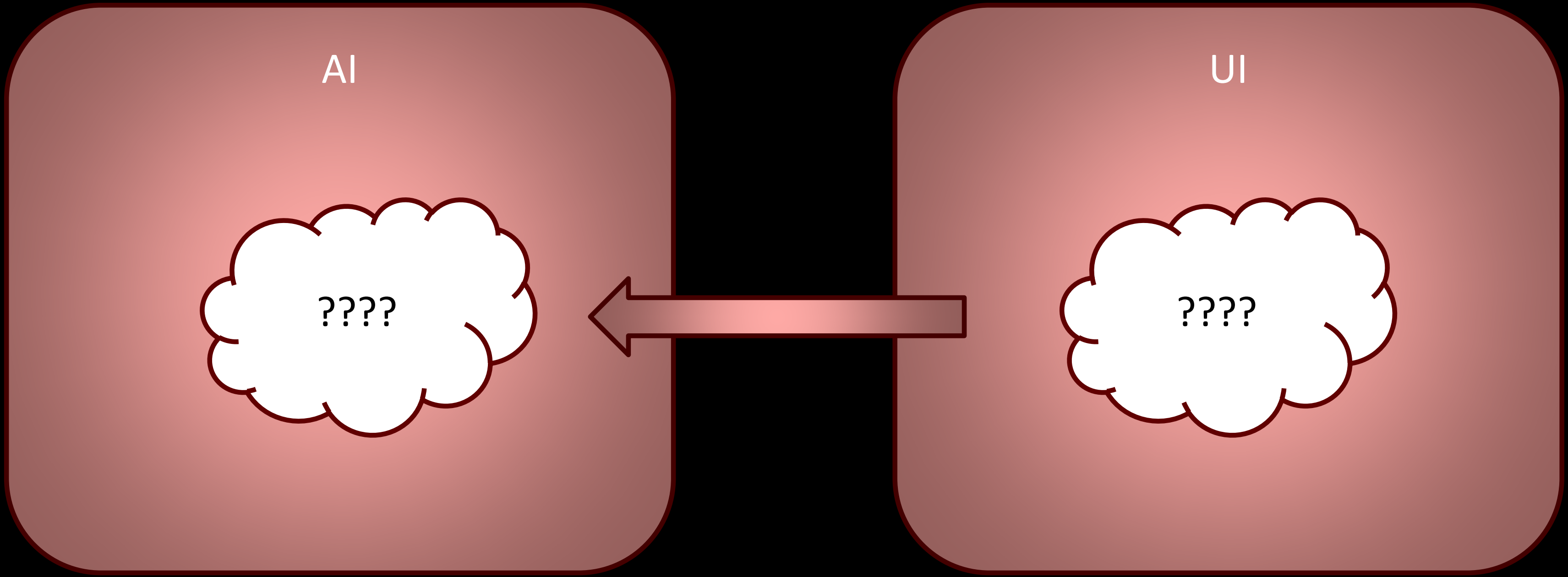
- Neutral
- Suspicious
- Alarmed
- Hostile



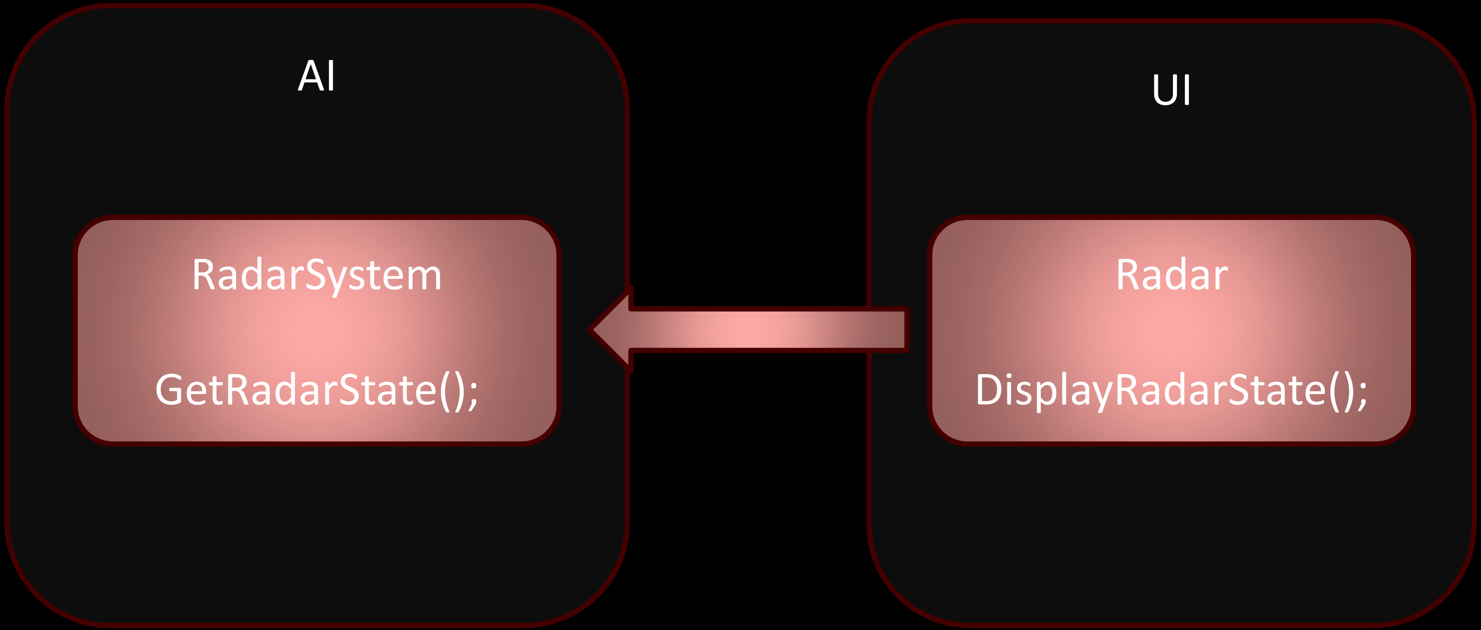
Radar

- Neutral
- Suspicious
- Alarmed
- Hostile

Case 2 : External Reasoning



Case 2 : External Reasoning



Case 2 : External Reasoning

- Lessons
  - Encapsulate the reasoning at only one location
  - The system owning the data is probably the best suited to own that reasoning

Case 3

Dealing With Archetypes

Case 3 : Dealing With Archetypes

Human

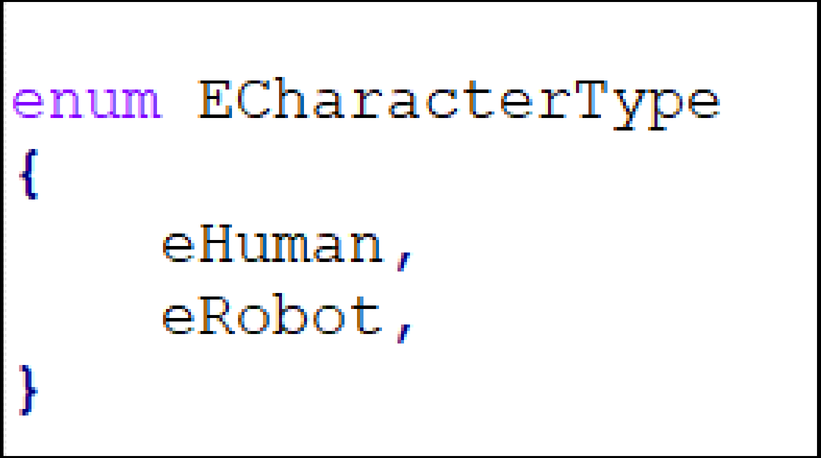
- Agile and intelligent
- Weak to poison
- Use cover

Robot

- Slow and resistant
- Weak to EMP
- Fights in the open

Case 3 : Dealing With Archetypes

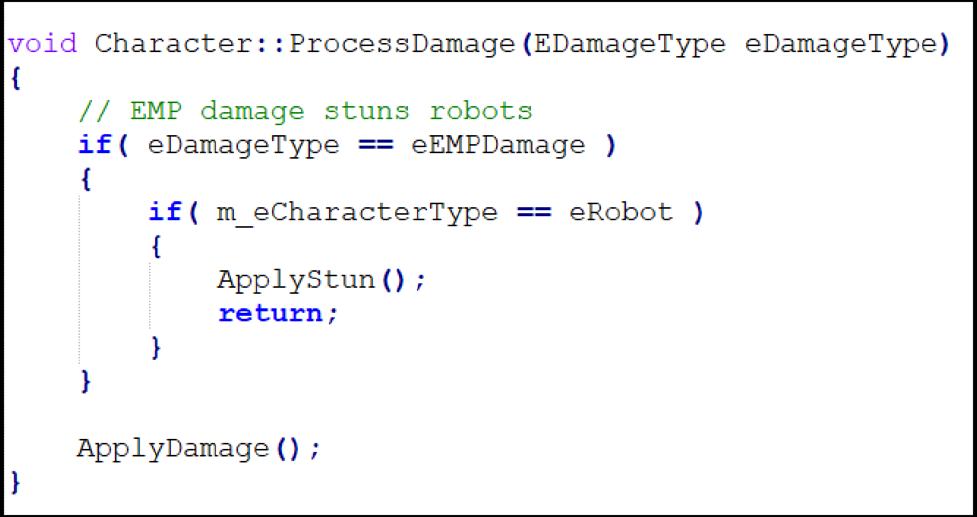
- Archetype becomes a tag in code



```
enum ECharacterType
{
    eHuman,
    eRobot,
}
```

Case 3 : Dealing With Archetypes

- Resolving stun damage



```
void Character::ProcessDamage(EDamageType eDamageType)
{
    // EMP damage stuns robots
    if( eDamageType == eEMPDamage )
    {
        if( m_eCharacterType == eRobot )
        {
            ApplyStun();
            return;
        }
    }

    ApplyDamage();
}
```

Case 3 : Dealing With Archetypes

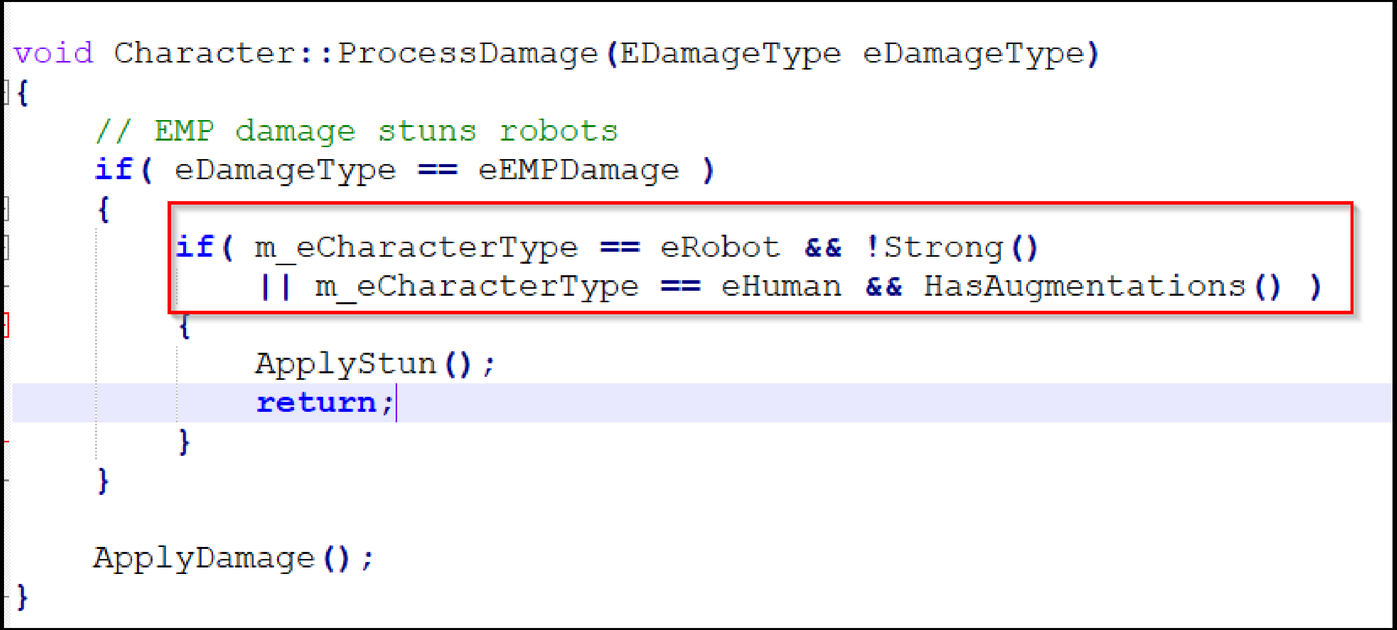
Augmented Human

- Is a human
- Is weak to EMP

Strong Robot

- Is a robot
- Immune to EMP

Case 3 : Dealing With Archetypes



```cpp
void Character::ProcessDamage(EDamageType eDamageType)
{
    // EMP damage stuns robots
    if( eDamageType == eEMPDamage )
    {
        if( m_eCharacterType == eRobot && !Strong()
            || m_eCharacterType == eHuman && HasAugmentations() )
        {
            ApplyStun();
            return;
        }
    }

    ApplyDamage();
}
```

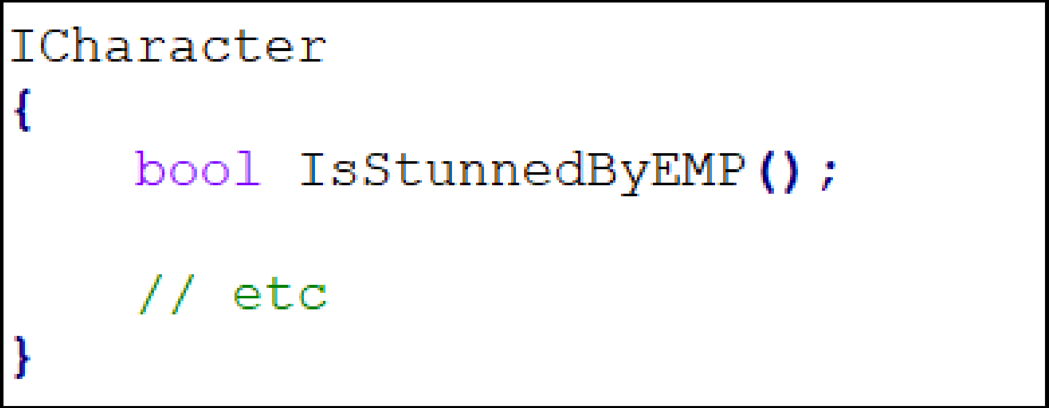
Case 3 : Dealing With Archetypes

- Conflict
  - Archetype design
  - Individual enemy properties

Case 3 : Dealing With Archetypes

- Solution
  - Define an interface for your characters
  - Query each property individually
  - Each character can be configured according to that property

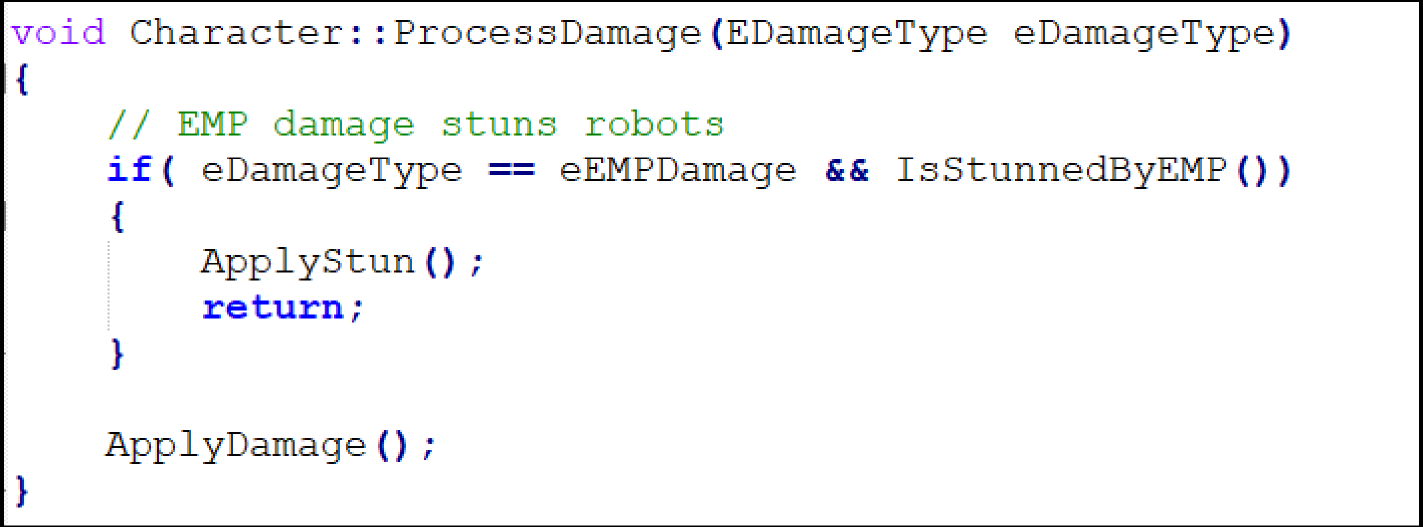
Case 3 : Dealing With Archetypes



```
ICharacter
{
    bool IsStunnedByEMP();

    // etc
}
```

Case 3 : Dealing With Archetypes



```
void Character::ProcessDamage(EDamageType eDamageType)
{
    // EMP damage stuns robots
    if( eDamageType == eEMPDamage && IsStunnedByEMP())
    {
        ApplyStun();
        return;
    }

    ApplyDamage();
}
```

Case 3 : Dealing With Archetypes

- IsPlayer()

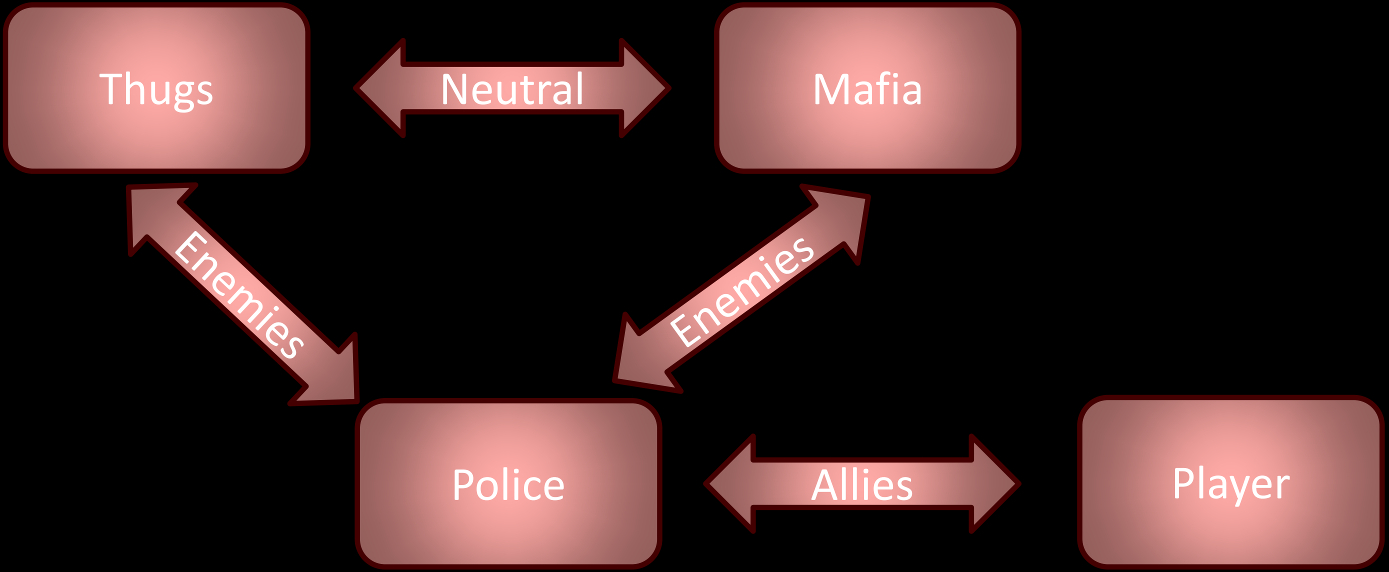
Player

```
IsPlayer()
{
    return true;
}
```

NPC

```
IsPlayer()
{
    return false;
}
```

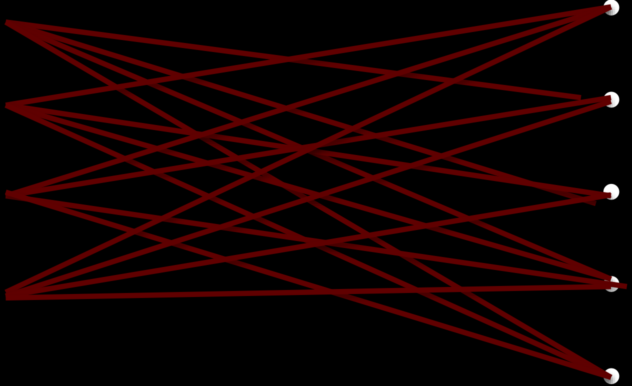
Case 3 : Dealing With Archetypes



Case 3 : Dealing With Archetypes

Factions

- Player
- Mafia
- Thugs
- Police
- etc...



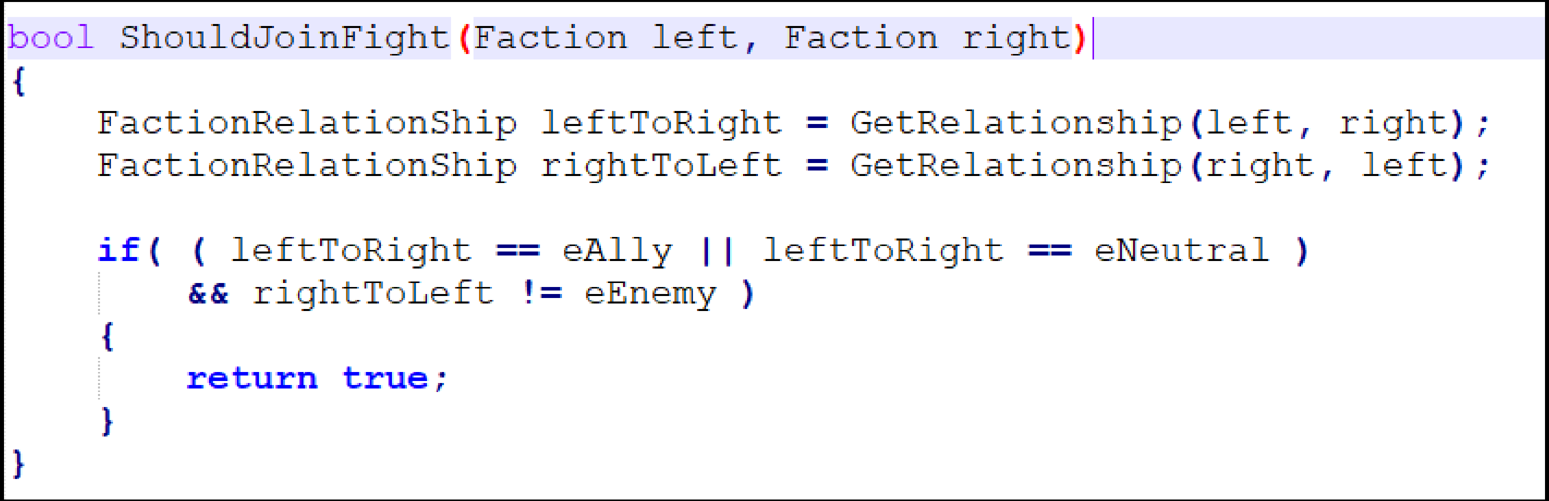
Factions

- Player
- Mafia
- Thugs
- Police
- etc...

Case 3 : Dealing With Archetypes

- Relationships
  - Allied
  - Friendly
  - Neutral
  - Wary
  - Enemy

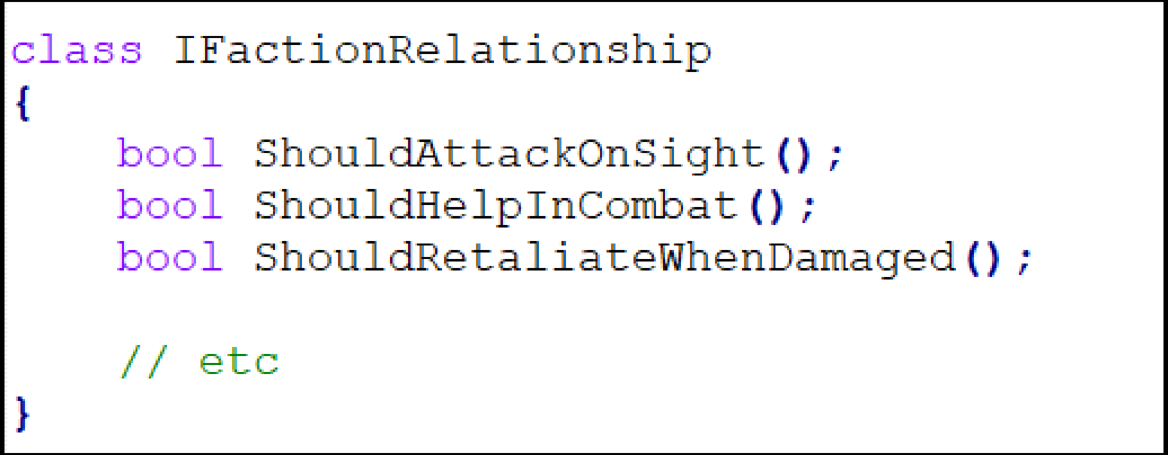
Case 3 : Dealing With Archetypes



```
bool ShouldJoinFight(Faction left, Faction right)
{
    FactionRelationShip leftToRight = GetRelationship(left, right);
    FactionRelationShip rightToLeft = GetRelationship(right, left);

    if( ( leftToRight == eAlly || leftToRight == eNeutral )
        && rightToLeft != eEnemy )
    {
        return true;
    }
}
```

Case 3 : Dealing With Archetypes



```
class IFactionRelationship
{
    bool ShouldAttackOnSight();
    bool ShouldHelpInCombat();
    bool ShouldRetaliateWhenDamaged();

    // etc
}
```

Case 3 : Dealing With Archetypes

- Lessons
  - Avoid linking groups of properties to archetypes, use properties individually
  - If tags are unavoidable, make them flexible through the use of an interface

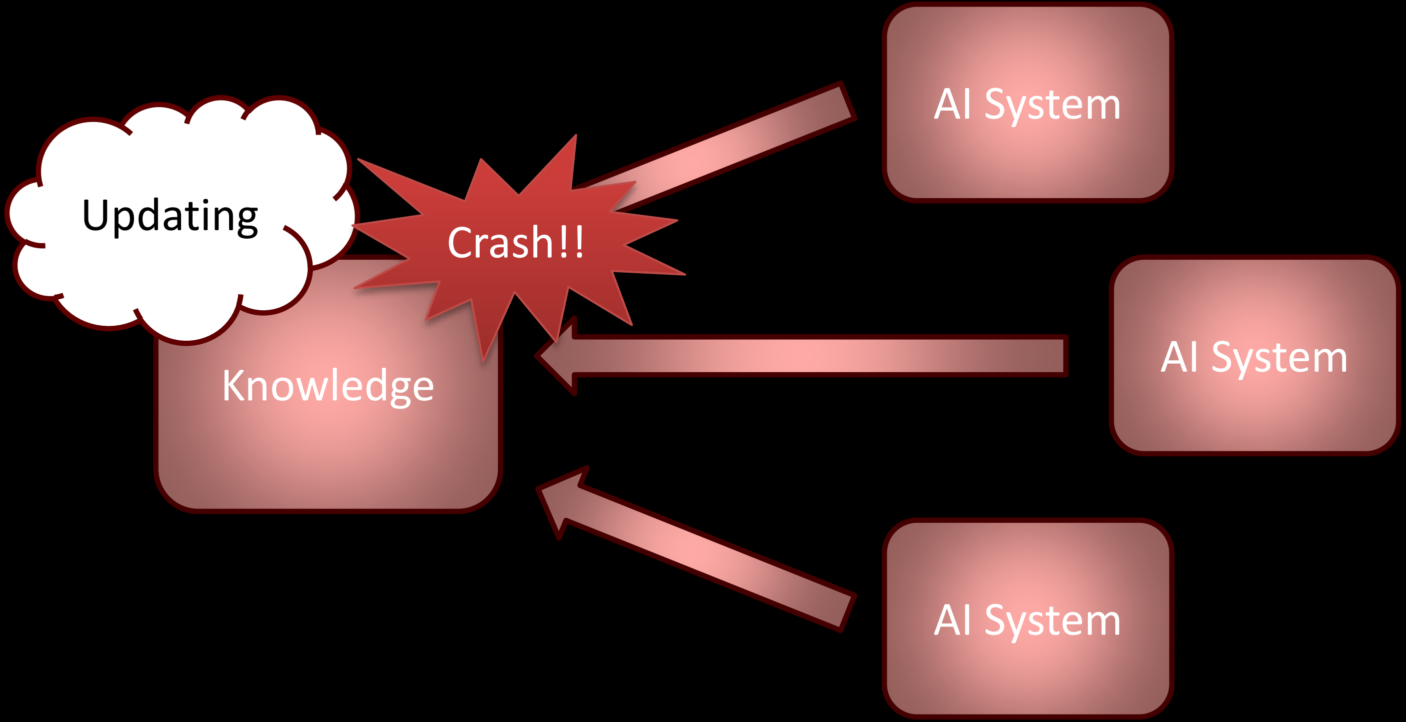
Case 4

Asynchronous Updates

Case 4 : Asynchronous Updates

- Main Thread
  - Worker Thread
  - Worker Thread
  - …

Case 4 : Asynchronous Updates

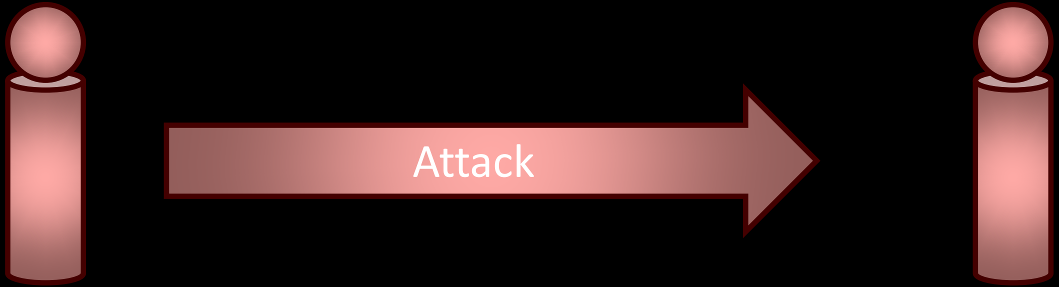


Case 4 : Asynchronous Updates

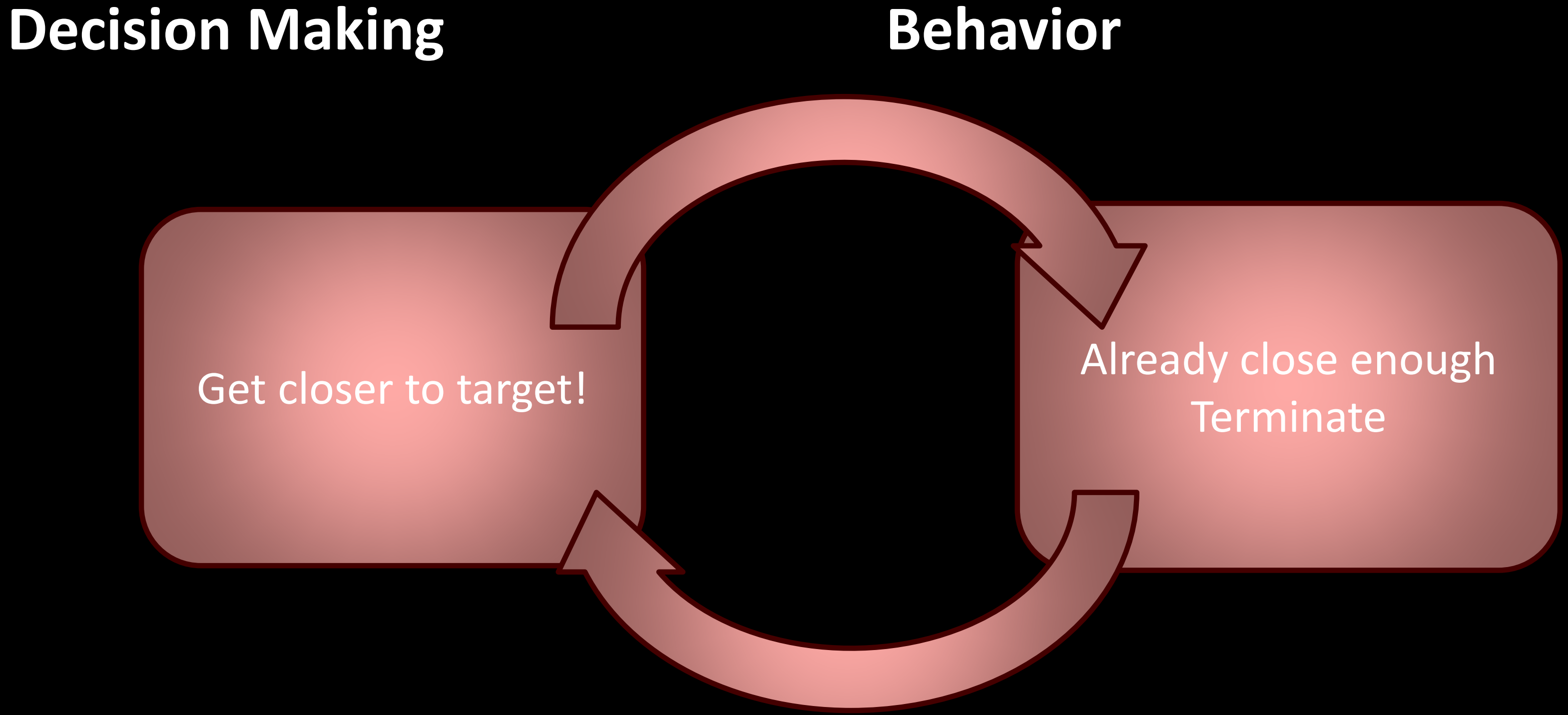
- Frame order
  - Access allowed
  - No access
    - Update knowledge
  - Access allowed

SAFE!

Case 4 : Asynchronous Updates

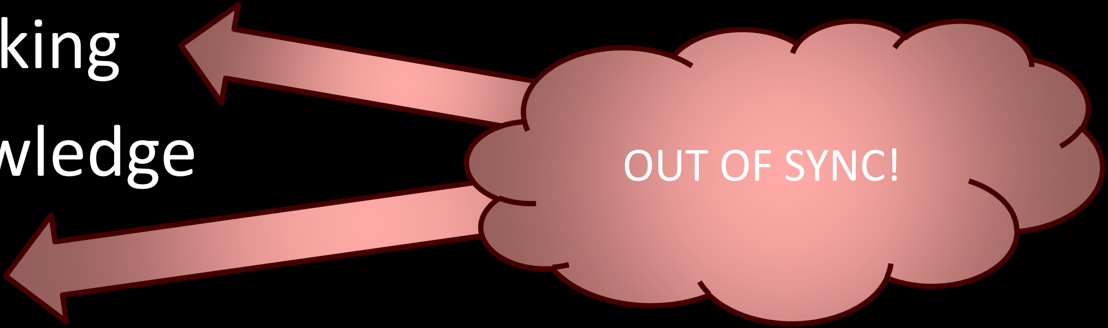


Case 4 : Asynchronous Updates



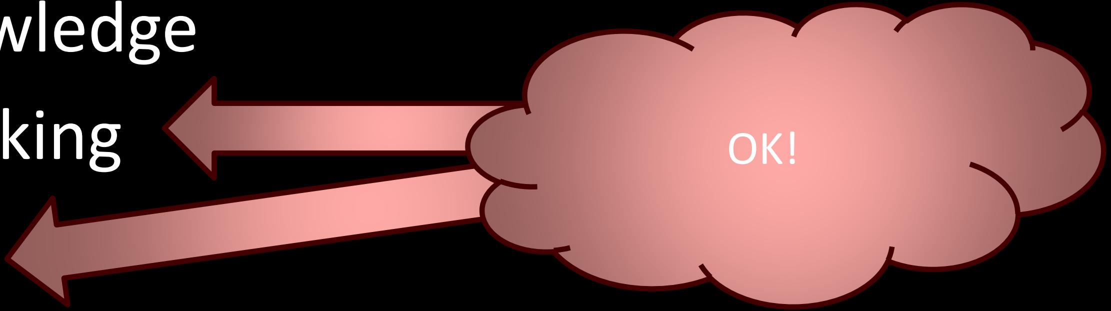
Case 4 : Asynchronous Updates

- Frame order
  - Decision making
  - Update knowledge
  - Execution



Case 4 : Asynchronous Updates

- Frame order
  - Update knowledge
  - Decision making
  - Execution

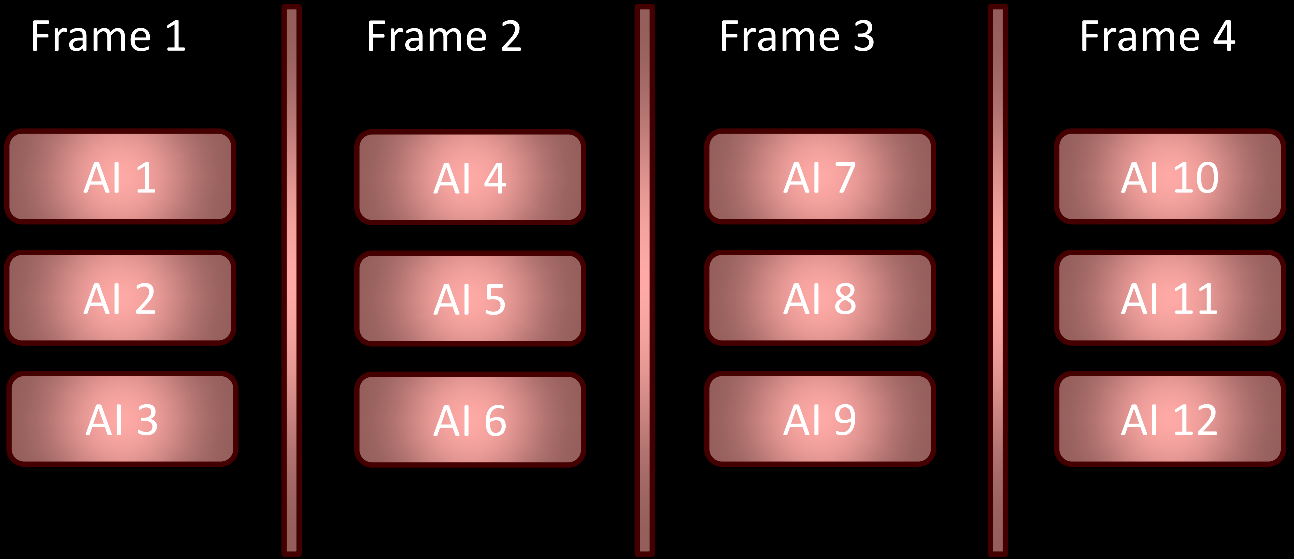


Case 4 : Asynchronous Updates

- Benefits of safe knowledge update
  - Time-slicing
  - Delegation architectures
  - Multi-frame requests

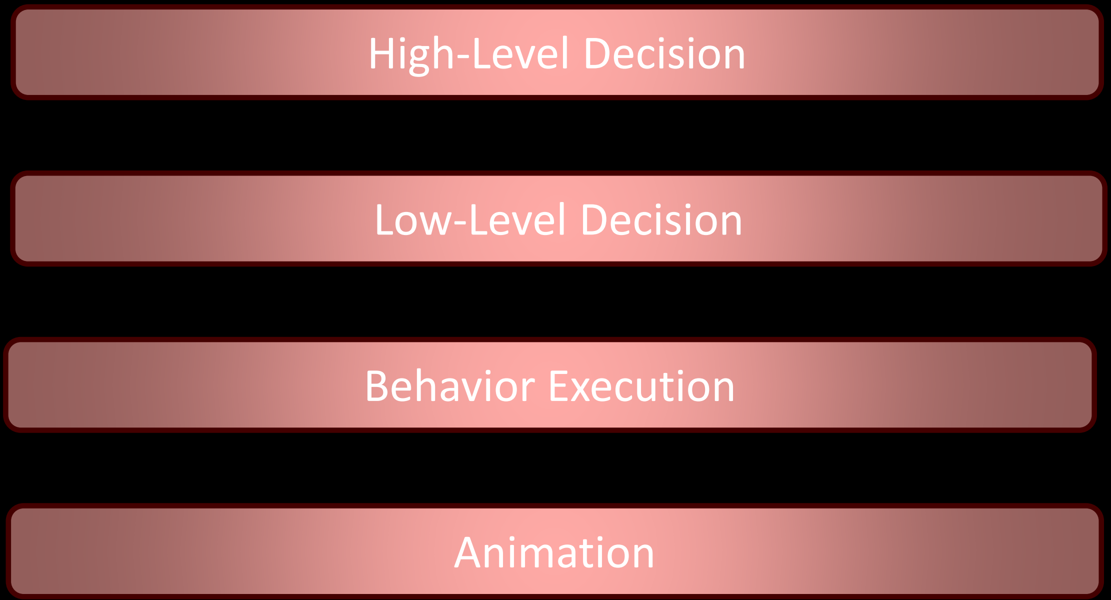
Case 4 : Asynchronous Updates

- Time slicing

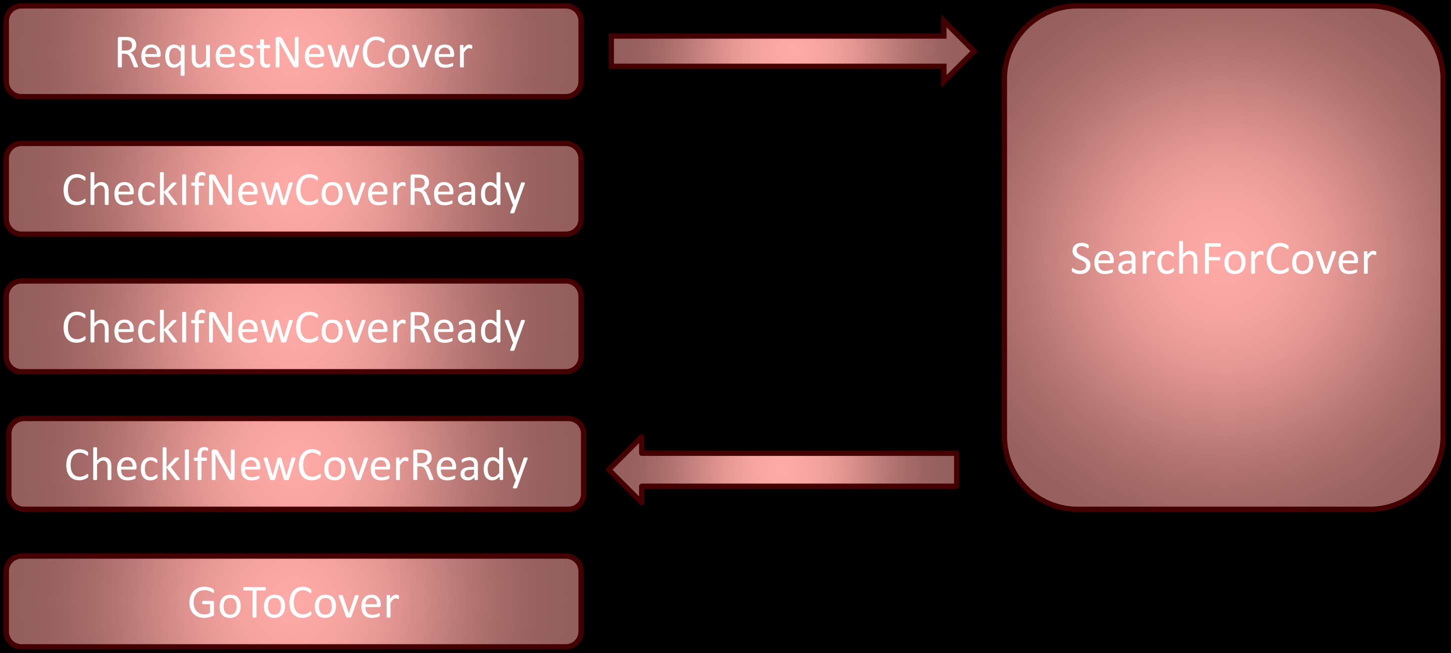


Case 4 : Asynchronous Updates

- Delegating Architecture



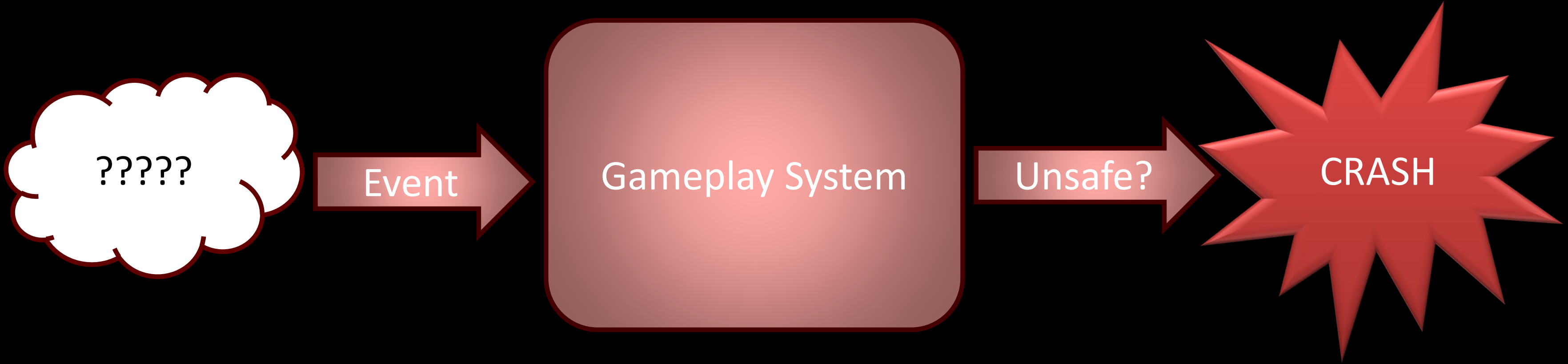
Case 4 : Asynchronous Updates



Case 4 : Asynchronous Updates

- Events
  - Easy to use interface
  - Reduce coupling between systems
  - Can communicate between code and data

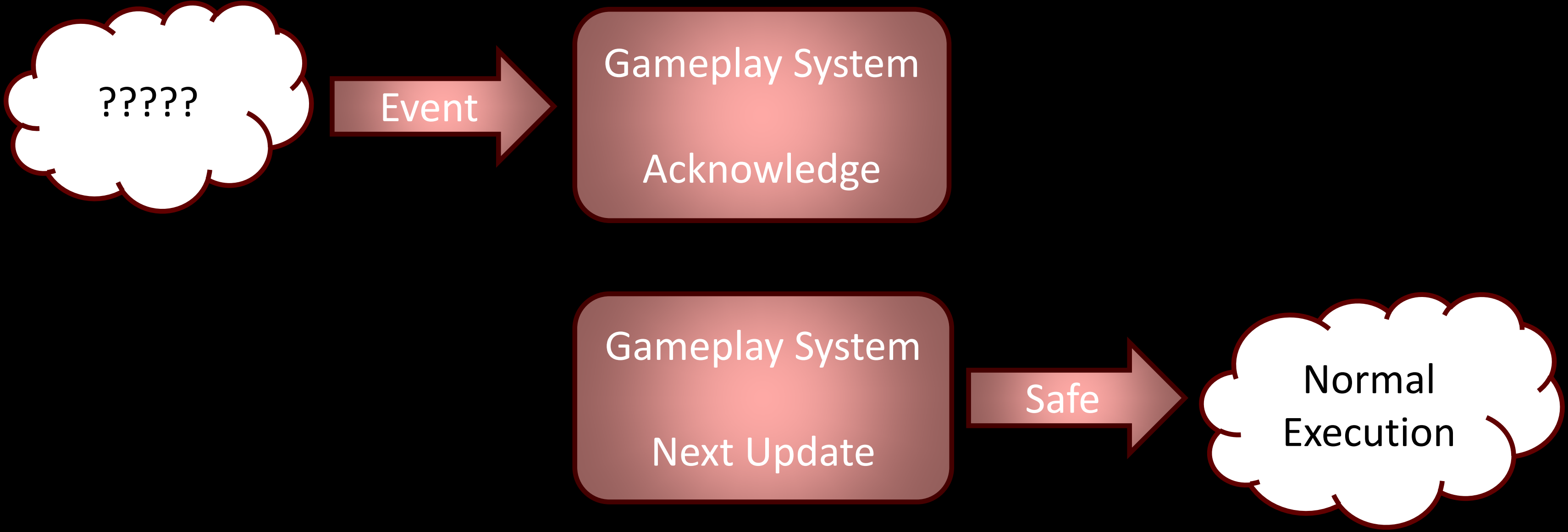
Case 4 : Asynchronous Updates



Case 4 : Asynchronous Updates

- Event Reception Flow
  - Receive Event
  - Acknowledge reception
  - Next main update: process events

Case 4 : Asynchronous Updates



Case 4 : Asynchronous Updates

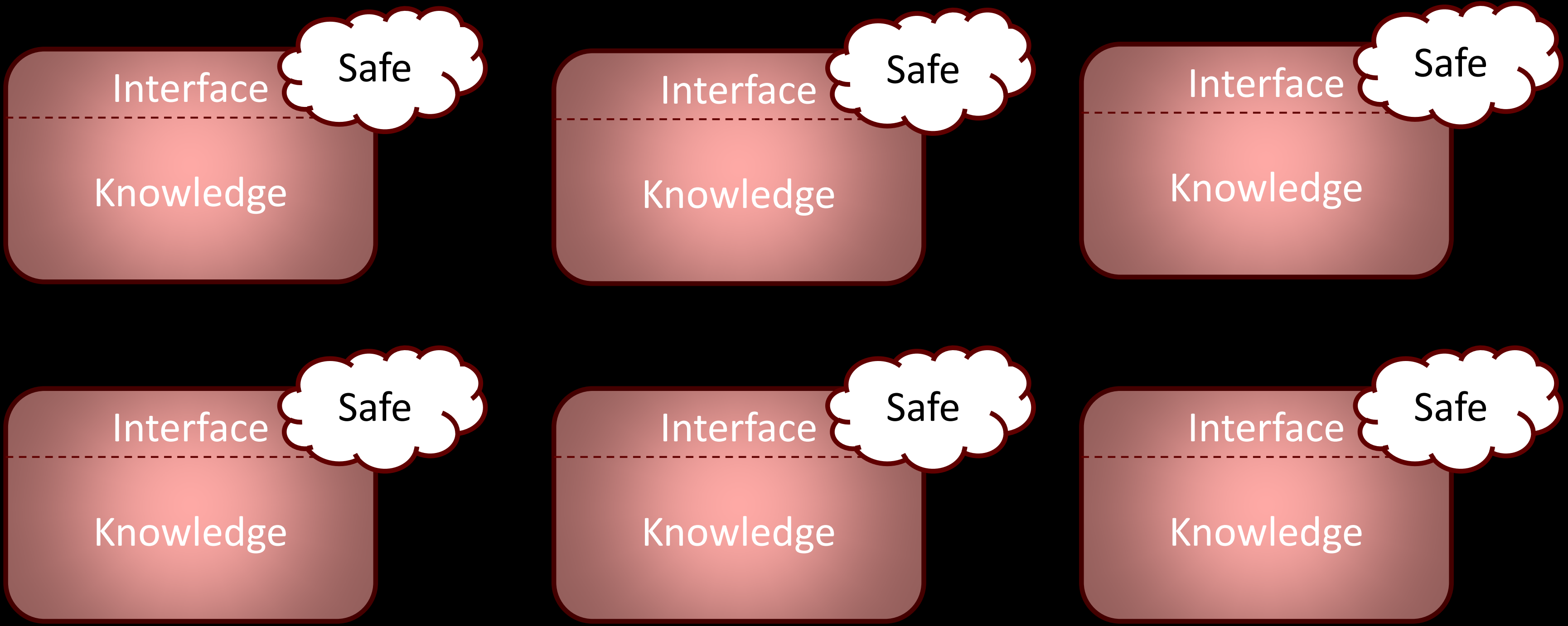
- Lessons
  - Update knowledge safely before accessing it
  - Safe knowledge update has multiple benefits
  - Acknowledge events and process them later

Putting It Together

Putting it all together

- Separate knowledge into small pods of specialized data
- Define clear interfaces offering reasoning on that specific knowledge
- Update safely before accessing

Putting it all together



Putting it all together

- Easy to understand
- Reasoning is left to the experts
- Safe updates lead to new benefits

Follow Up

Follow-up : Data Driven Content

CODE

VS

DATA

Follow-up : Data Driven Content

- Data driven is just another way to represent code
- Coding principles should be applicable
  - Divide into small groups that make sense
  - Create building blocks for specialized interfaces
  - Make sure update is safe

Follow Up : Prototyping vs Production

- Safer = more restrictions
  - Limited access
  - Specific organization

Follow Up : Prototyping vs Production

- When prototyping
  - Ok to have less restrictions
  - Goal is to go fast
- When production
  - Time to make things more robust
  - Goal is to be safe

Follow Up : Improved Communication

- Restrictions mean people need to talk
  - Problems are explained explicitly
  - Experts can figure out the best solution
  - Less solitary struggles

Final Thoughts

- Naming is important
- Identify what is painful
- There is no perfect solution

Conclusion

- Knowledge is important
- Consider how your knowledge flows through your architecture
- Try to reduce the amount of knowledge in one place
- Use interfaces to let expert systems do the reasoning
- Make sure your knowledge update is safe, and use asynchronous updates
- Figure out what is best for your team

Questions