A Simple Rollback System in C++

The Secret Behind Online Multiplayer Games

ELIAS FARHAN

Summary

- Introduction
- Why is it hard?
- Why deterministic simulation?
- What is rollback?
- Implementation details
- Debugging
- Improvements

Elias Farhan

- Head Instructor of Games Programming at SAE Institute Geneva
- Co-founder of indie game studio Team KwaKwa
- Co-organizer of the Suisse Romande Game Dev Meetup

Splash Blast Panic

Doing fast-paced online multiplayer, the wrong way…

The wrong way…

```
stompPrepTimer.Reset();
stompPrepared = true;
GetComponent<PlayerFX>().dashPrepFX();
hasPrep = true;
stopDash = false;
stopDashTimer.Stop();
slowDashTimer.Stop();
dashedTimer.Stop();
dashed = false;

doubleDashed = false;
dashing = false;
if (!dashing)
    soundManager.PlaySound(PlayerSoundManager.PlayerSound.STOMPPREP,
                           SoundManager.PlayerPitch.RANDOM);
```

Mixing spawning visual fx, sound and gameplay code…

Splash Online (2024)

C++-20 reimplementation

Using:

- SDL2
- Spine
- Photon
- Fmod
- DearImgui

Available here!

One code example

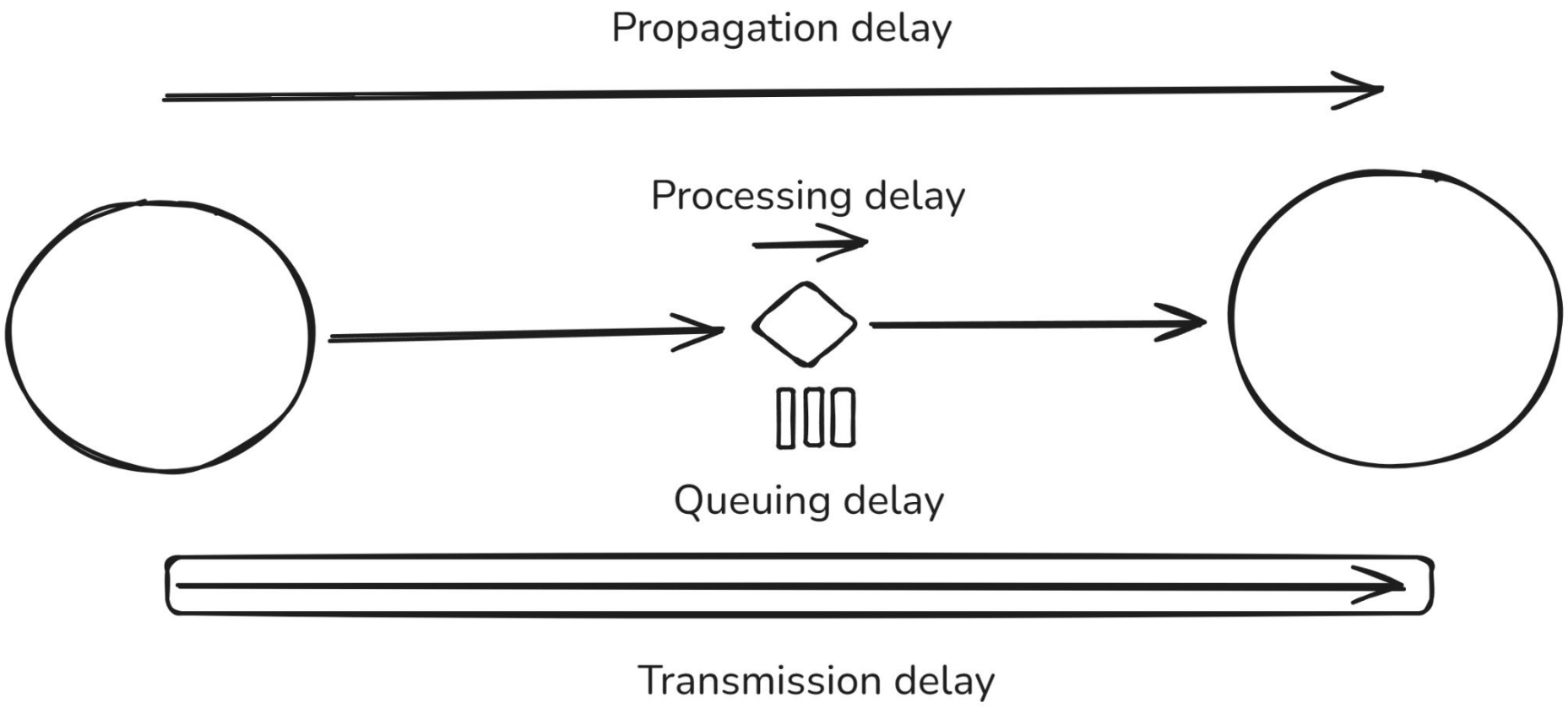
```
[System.Serializable]
public class Chronometer
{
    public float period;
    public float time;
    public void Update(float deltaTime){...}
    public void Reset(){...}
    public bool Over(){...}
```

```
template<typename T=neko::Scalar,
    int Period=0,
    int Dividend=1000>
class Timer
{
public:
    static constexpr T period{(T)Period/(T)Dividend};
```

SAE INSTITUTE

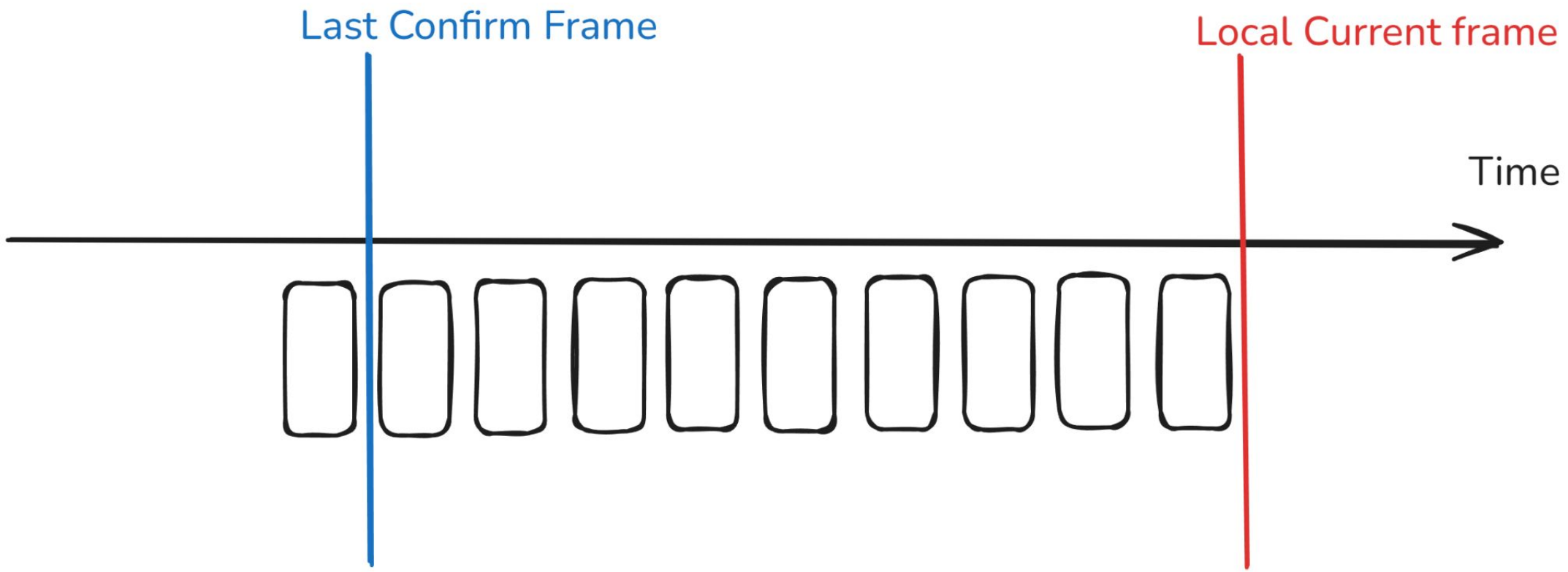
Why is it harder to make a fast-paced online game?

Network Latency

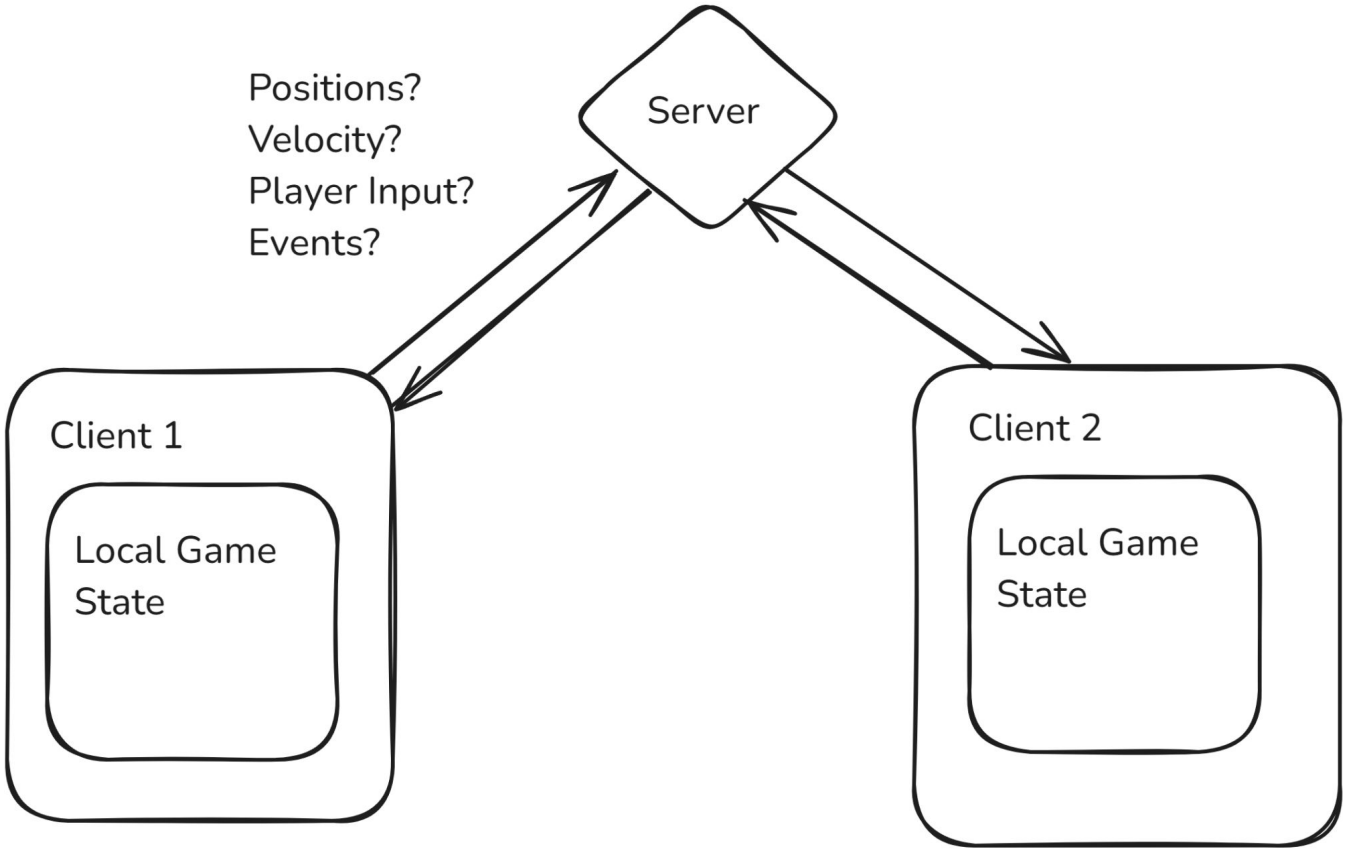


It means we are waiting for other player inputs…

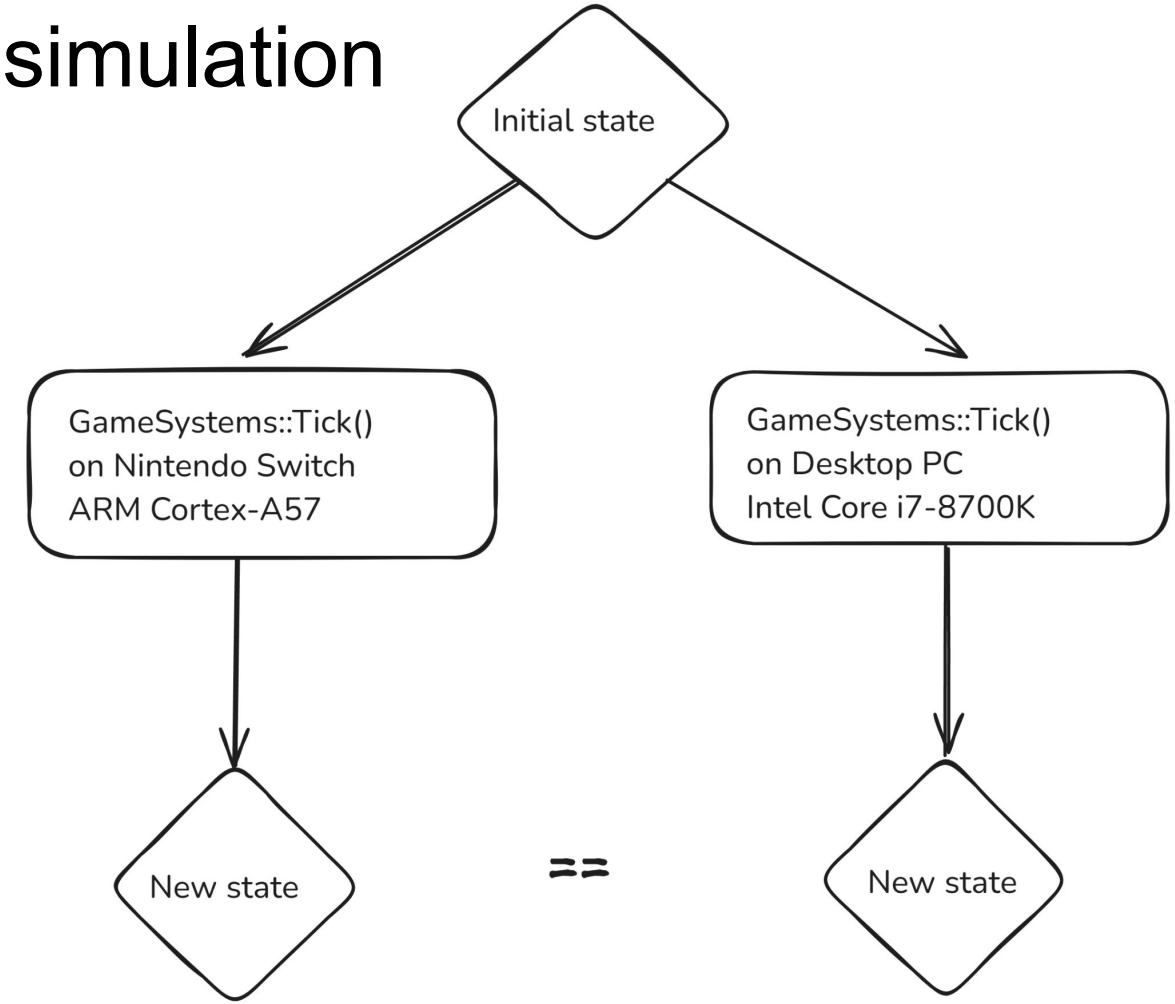
Confirm frame and current frame



Data sent through the network



Deterministic simulation

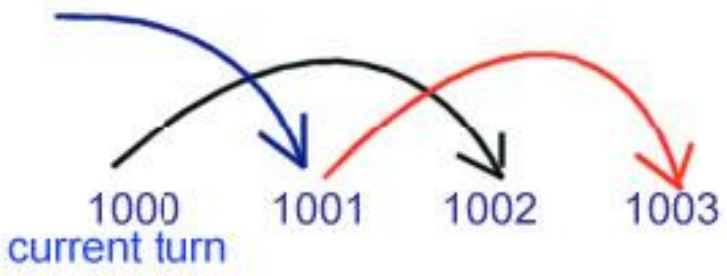


With a deterministic simulation, we mostly need to send player inputs!

Deterministic lock-step

For example in *Age of Empires*.

From the article by Paul Bettner: *1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond*

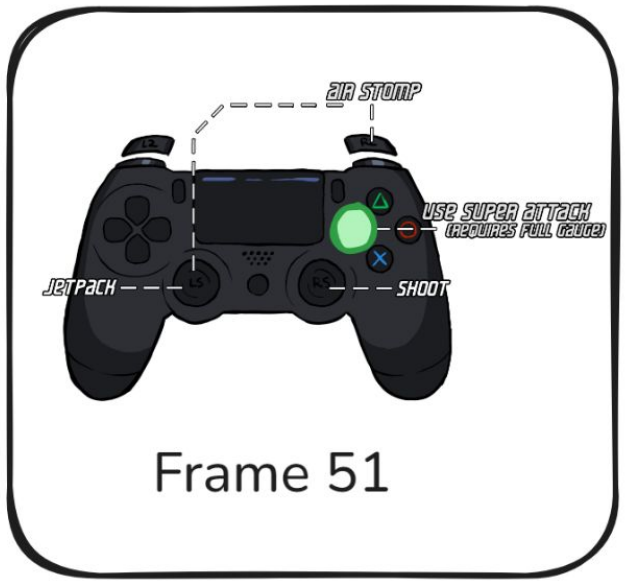


We still need to wait for all the player inputs to process the confirm frame…

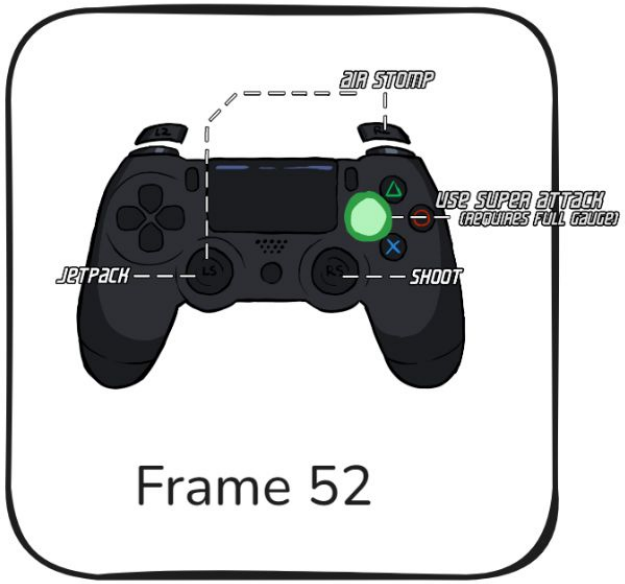
What is a rollback system?

Input prediction

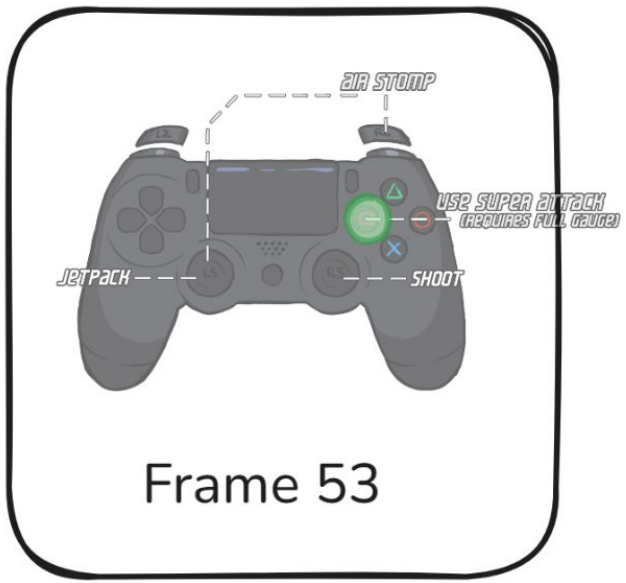
Last received input



Frame 51

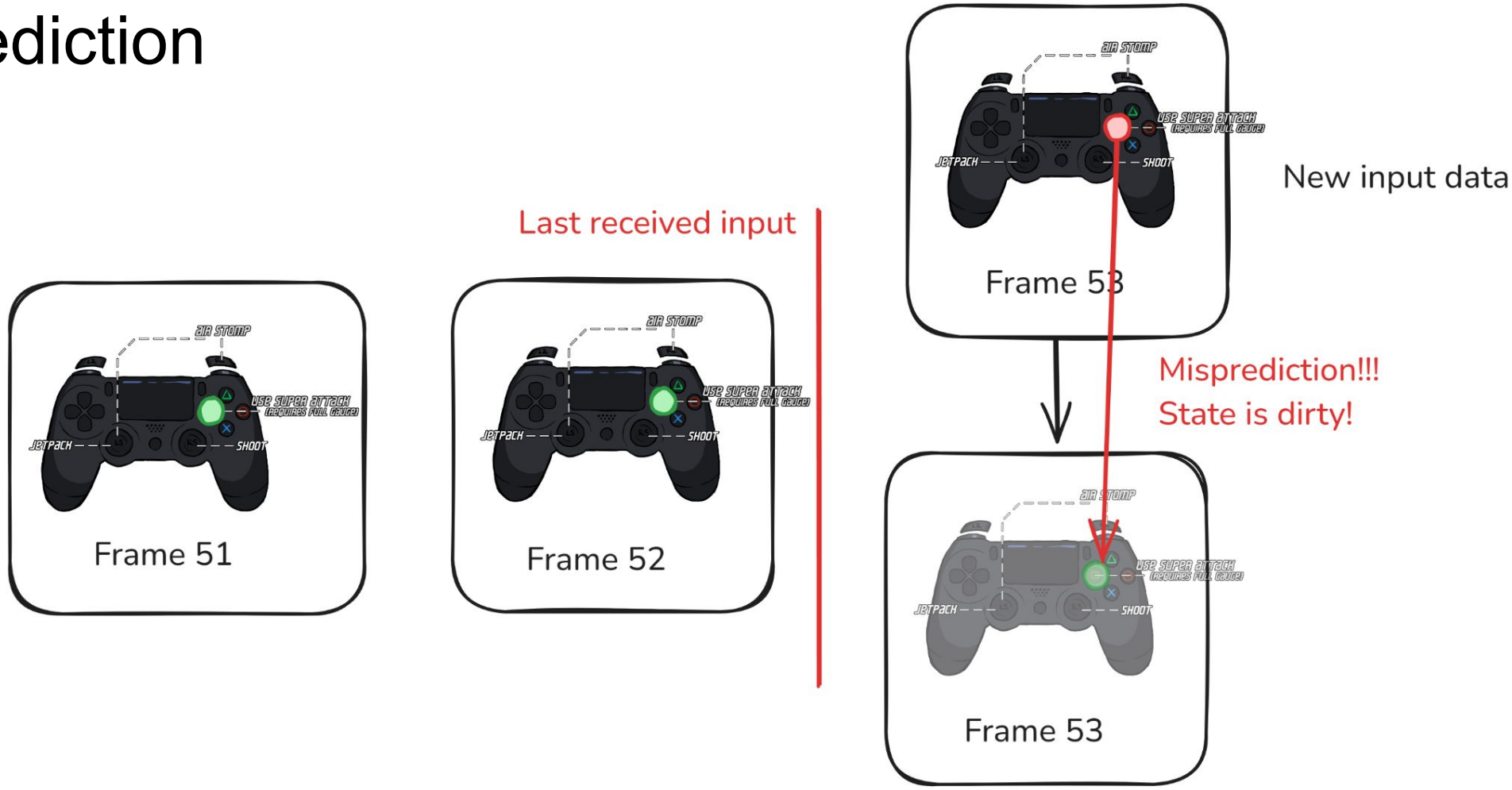


Frame 52

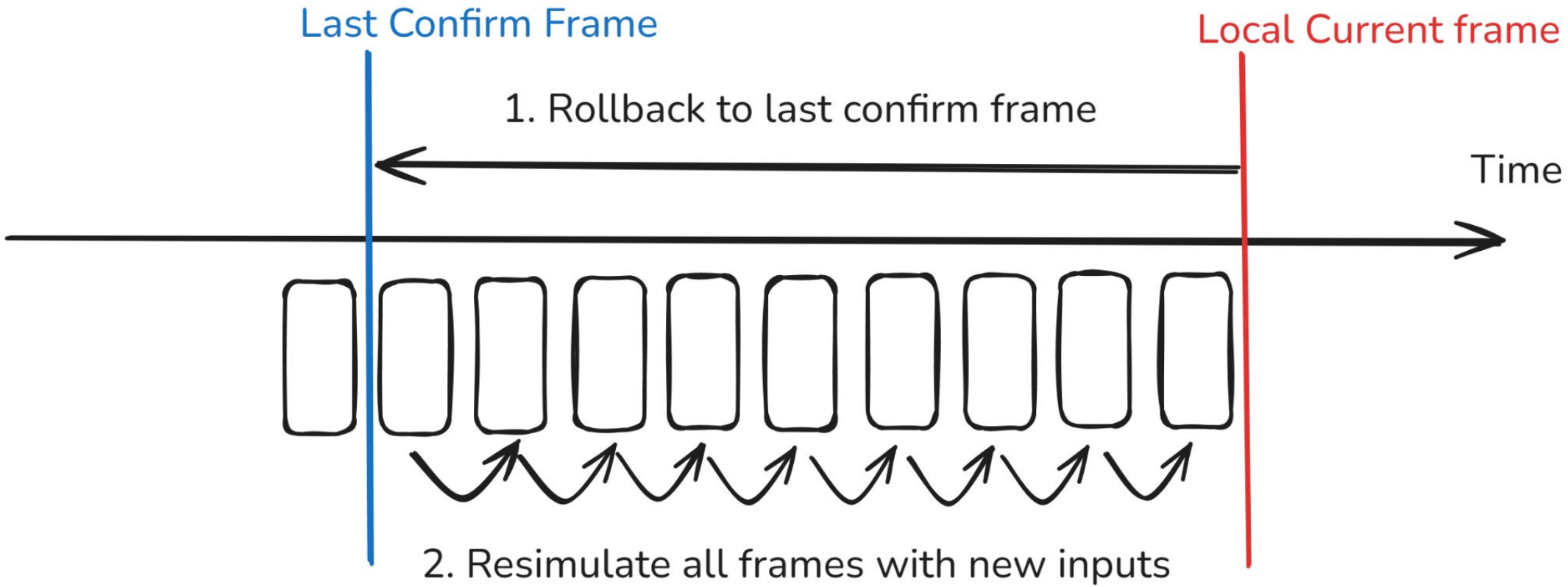


Frame 53

Misprediction



Doing a rollback

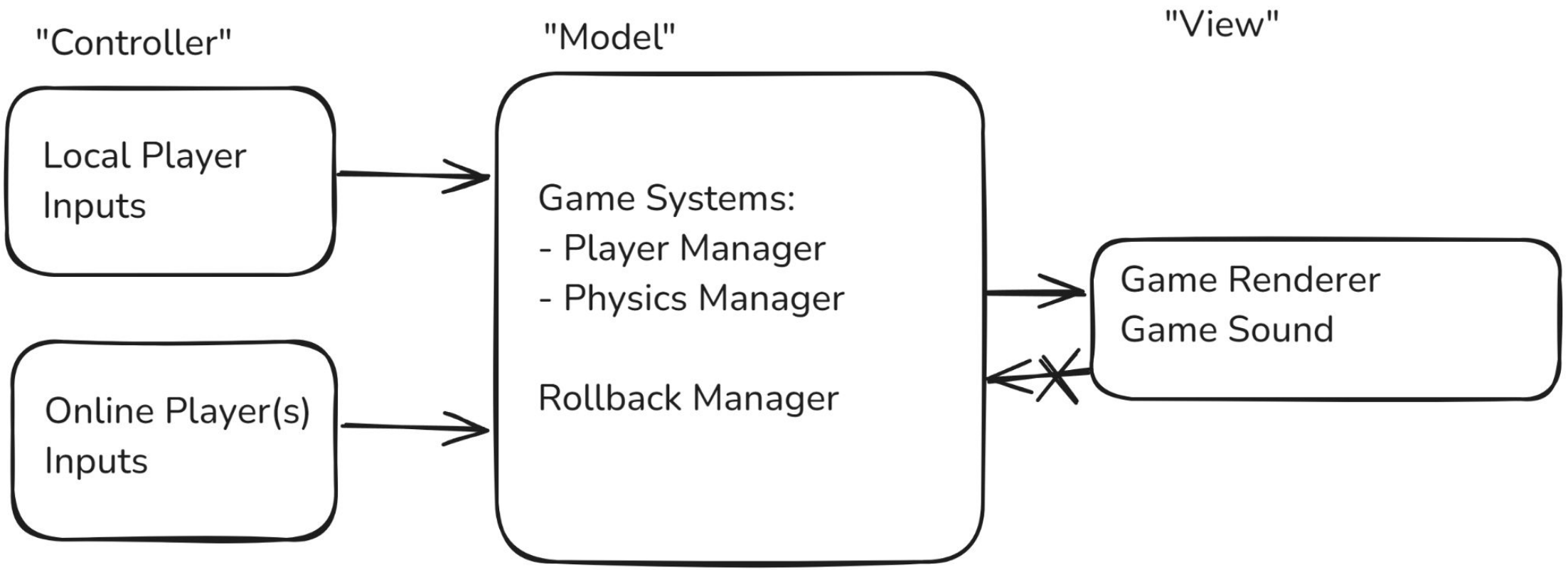


Rollback Implementation Details

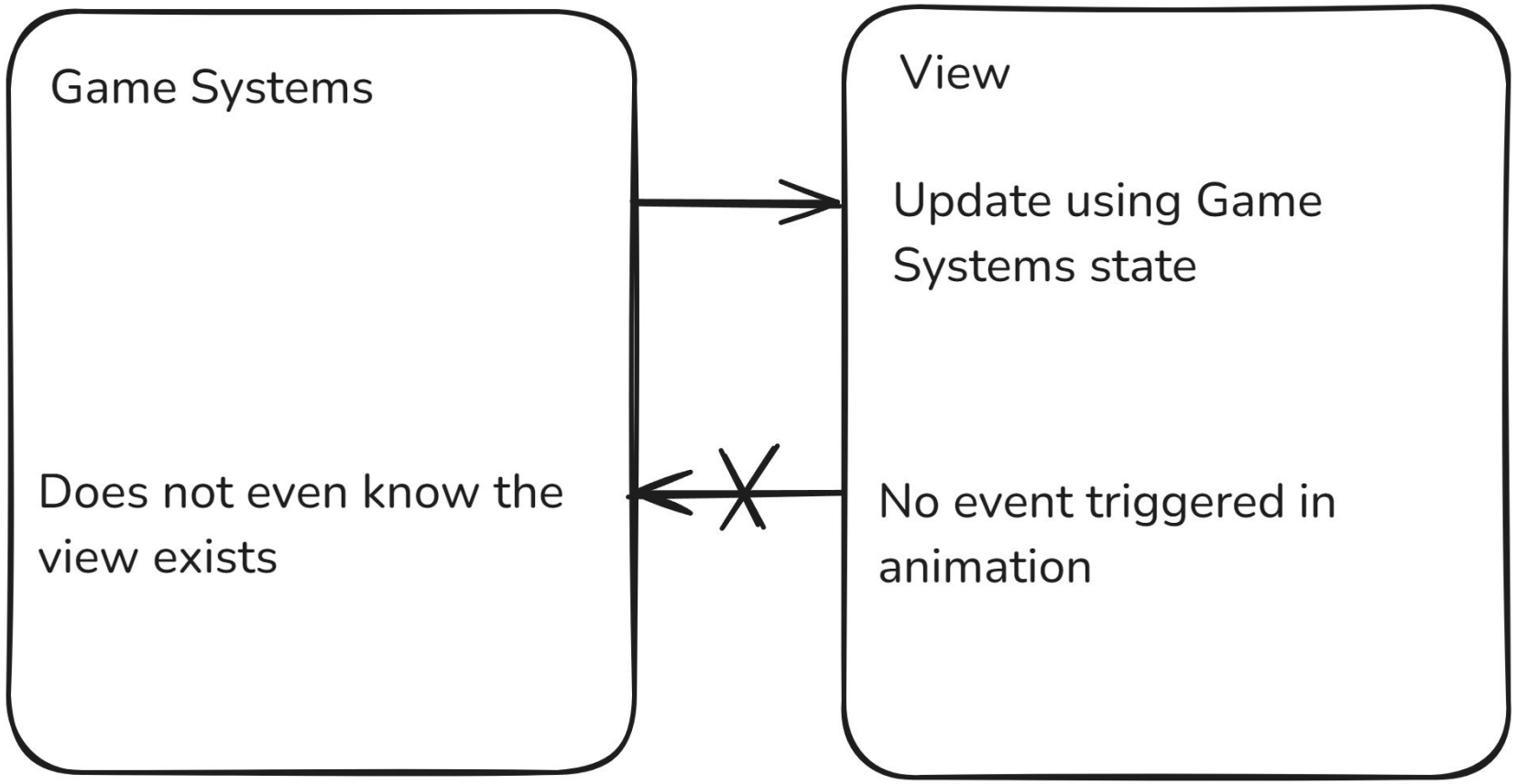
Physics vs Graphics game state

Parallel Universes in SM64 by pannenkoek2012 in *Watch for Rolling Rocks - 0.5x A Presses (Commentated) [OUTDATED]*

MVC-like architecture



View


Game Systems
Does not even know the view exists
View
Update using Game Systems state
No event triggered in animation

Fixed timestep

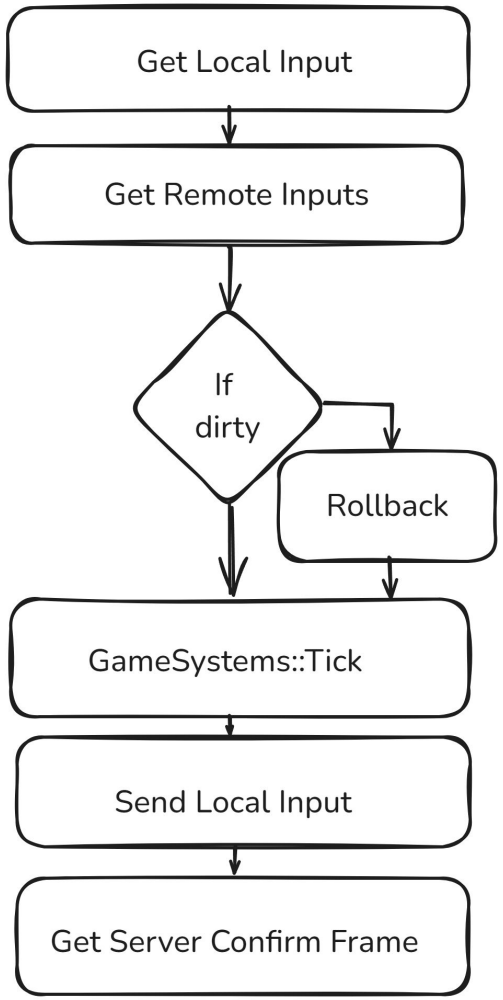
From [Fix Your Timestep!](#) by Glenn Fiedler:

Separate your game systems update from your view update.

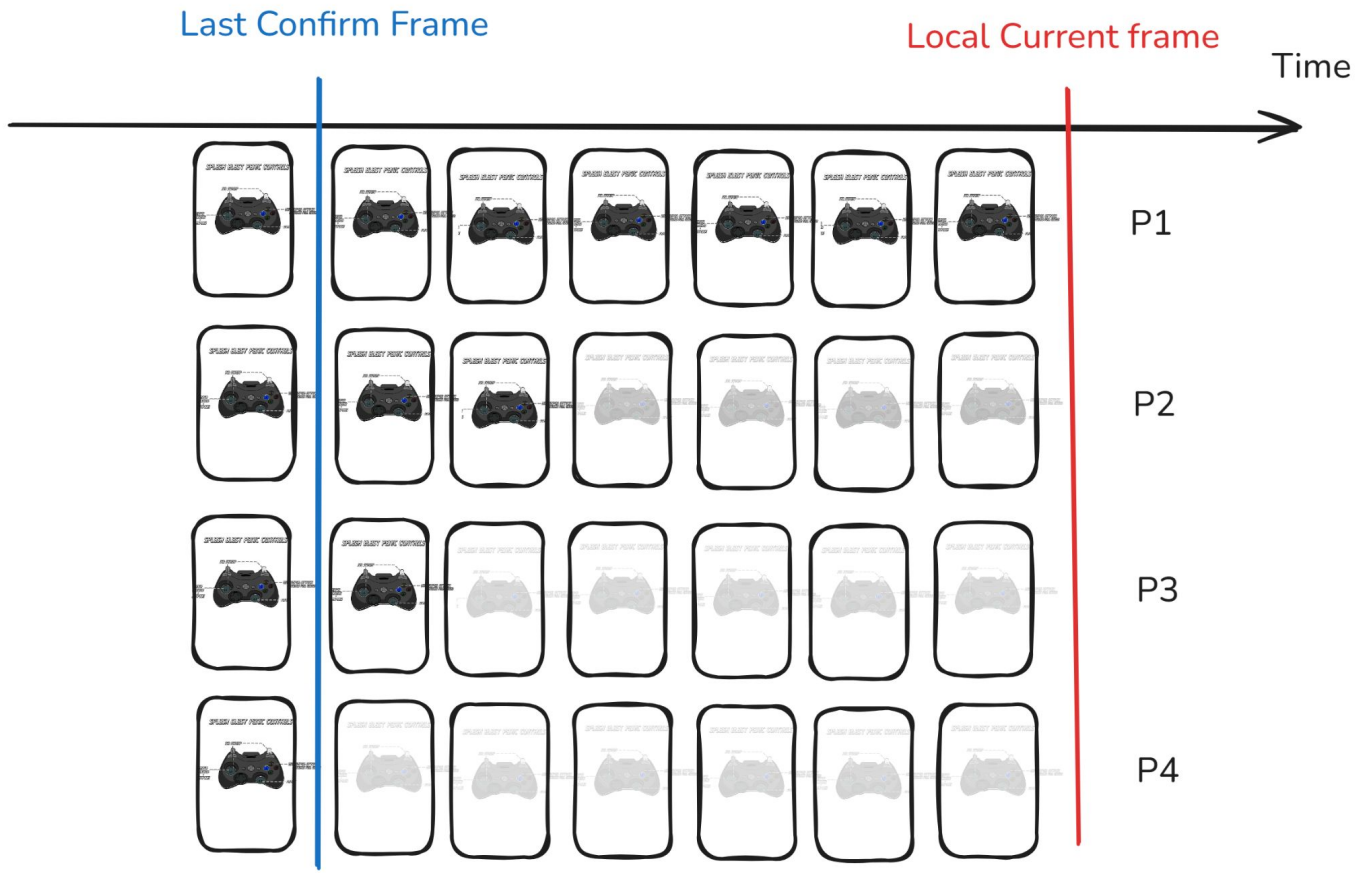
This allows to sequence your game frame.

```
//In Update
currentTime_ += dt;
while (currentTime_ > fixedDt)
{
    Tick();
    currentTime_ -= fixedDt;
}
```

A Fixed Tick


Get Local Input
Get Remote Inputs
If dirty
Rollback
GameSystems::Tick
Send Local Input
Get Server Confirm Frame

Rollback Manager Inputs



Rollback

```
if(rollbackManager_.IsDirty())
{
    gameSystems_.RollbackFrom(rollbackManager_.GetGameSystems());
    for(int i = 0; i < currentFrame_-rollbackManager_.GetLastConfirmFrame(); i++)
    {
        const auto rollbackInputs = rollbackManager_.GetInputs(firstFrame+i);
        gameSystems_.SetPlayerInput(rollbackInputs);
        gameSystems_.Tick();
    }
    rollbackManager_.SetDirty(false);
}
```

Game System Architecture

Game systems need a function to rollback (revert to a previous state).

Typically a simple copy of data.

```cpp
template<typename T, //...>
class RollbackInterface
{
public:
    virtual void RollbackFrom(const T& system) = 0;
    //....
};
```

```cpp
void BulletManager::RollbackFrom(const BulletManager& system)
{
    bullets_ = system.bullets_;
}
```

Naïve OOP architecture

- Hard to copy
- Hard to serialize
- Pointers everywhere

Player Character as a gameplay, physics, graphics and audio object…

```
class GameObject
{

public:
    virtual void Begin() = 0;
    virtual void Update(float dt) = 0;
    virtual void Draw() = 0;
    virtual void End() = 0;
private:
  Vec2f position_;
};

class PlayerCharacter : public GameObject
{
  //...
};

class Bullet : public GameObject
{
  //...
};


std::vector<std::unique_ptr<GameObject>> gameObjects_;
```

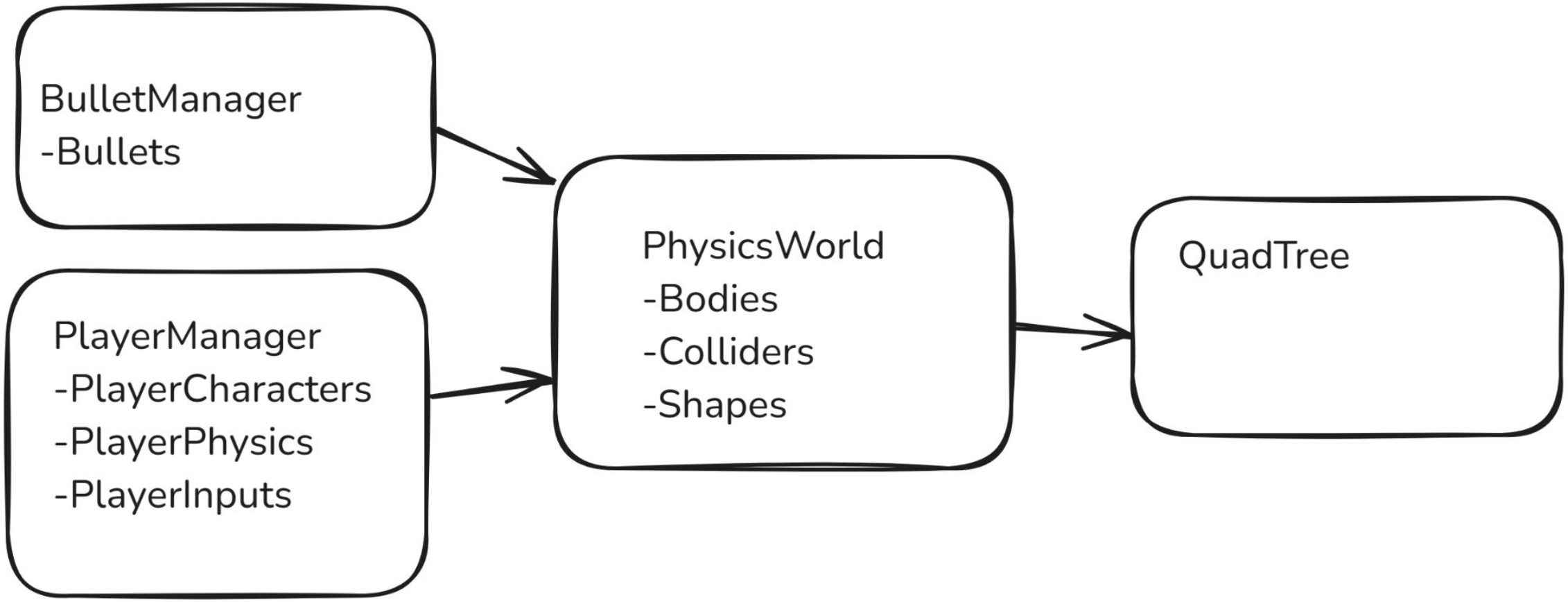
ECS-like architecture

System and array of Components (C-struct without functionality, except operators maybe)

You want to have a manager of PlayerCharacter, not a PlayerCharacter script.

```
class PlayerManager
{
public:
  void Tick()
  {
    for(auto& playerCharacter: playerCharacters_)
    {
      //Same as PlayerCharacter::Tick();
    }
  }

  void Rollback(const PlayerManager& other)
  {
    playerCharacters_ = other.playerCharacters_;
  }
private:
  std::vector<PlayerCharacterStruct> playerCharacters_;
};
```

Splash Online Game Systems



Physics Engine

1. Try not to use a physics engine… (like Beach Slap)
2. Make your own physics engine featuring internal state copy!

Rocket League uses a modified version of Bullet3.

Use index as pointer

Deterministic indices creation.

If you create three bodies with two copies of the PhysicsWorld, they will have the same BodyIndices.

```
class PhysicsWorld
{
public:
    BodyIndex AddBody();
    void RemoveBody(BodyIndex index);

    [[nodiscard]] Body& body(BodyIndex index);
    [[nodiscard]] const Body& body(BodyIndex index) const;

    ColliderIndex AddCircleCollider(BodyIndex body);
    ColliderIndex AddAabbCollider(BodyIndex body);
    ColliderIndex AddPlaneCollider(BodyIndex body);

    void RemoveAabbCollider(ColliderIndex index);
    void RemoveCircleCollider(ColliderIndex index);
    void RemovePlaneCollider(ColliderIndex index);

    [[nodiscard]] Collider& collider(ColliderIndex colliderIndex);
    [[nodiscard]] const Collider& collider(ColliderIndex colliderIndex) const;

    [[nodiscard]] AabbCollider& aabb(ShapeIndex shapeIndex);
    [[nodiscard]] const AabbCollider& aabb(ShapeIndex shapeIndex) const;
    [[nodiscard]] CircleCollider& circle(ShapeIndex shapeIndex);
    [[nodiscard]] const CircleCollider& circle(ShapeIndex shapeIndex) const;
    //...
};
```

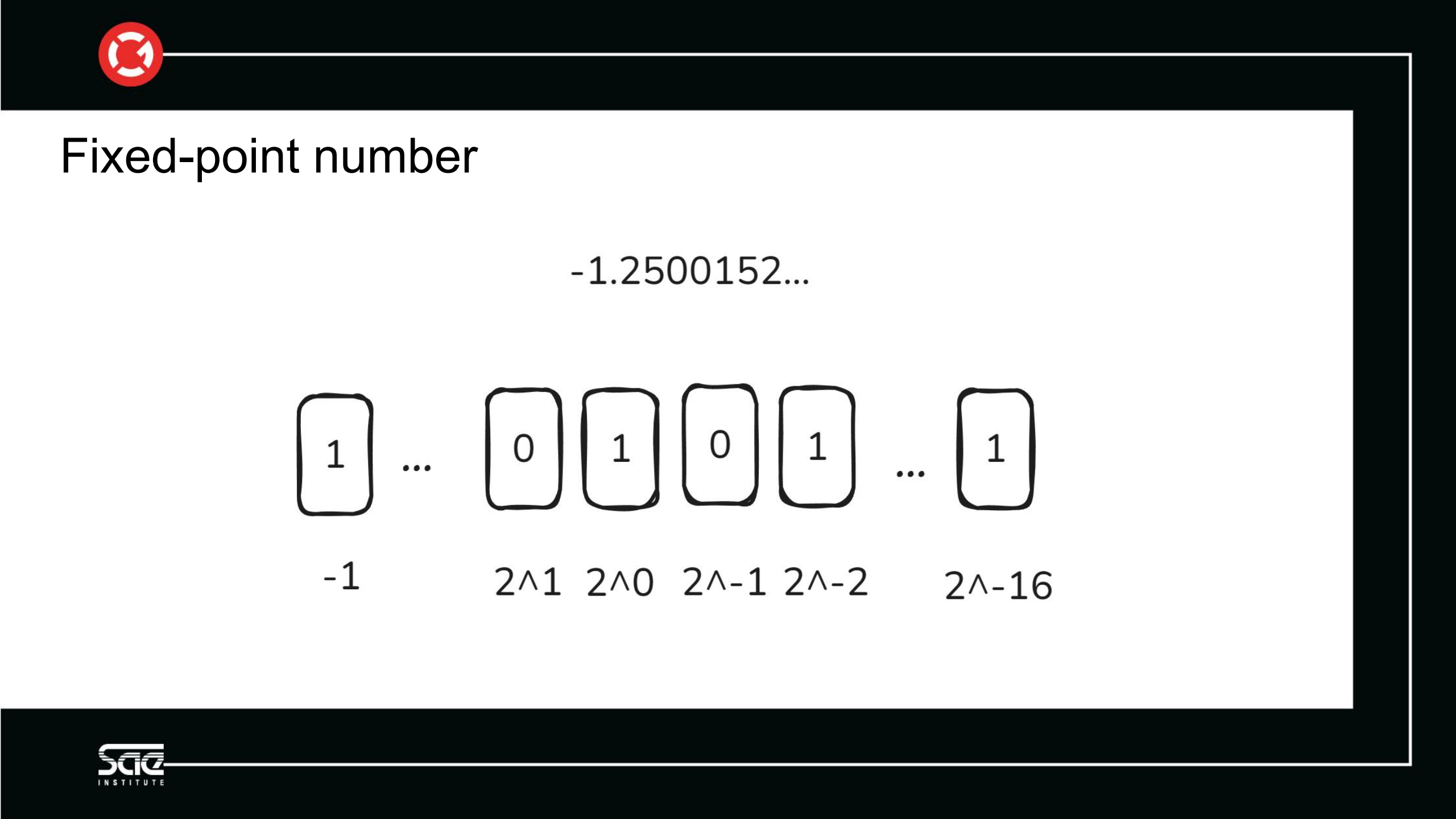
Floating-point number determinism

From Yossi Kreinin - *Consistency: how to defeat the purpose of IEEE floating point*

1. Algebraic compiler optimizations
2. "Complex" instructions
3. x86-specific pain.

SG14: P3375R0 tries to address this issue.

-> Talk by Sherry Ignatchenko tomorrow on this subject!



Fixed-point number

-1.2500152...

1 ... 0 1 0 1 ... 1

-1 2^1 2^0 2^-1 2^-2 2^-16

Fixed-point number

```cpp
constexpr explicit Fixed(float f)
{
    underlyingValue_ = static_cast<std::int32_t>((f) * (1 << Exp));
}

constexpr explicit Fixed(std::int32_t i)
{
    underlyingValue_ = i * (1 << Exp);
}

constexpr Fixed operator+(Fixed other) const
{
    Fixed result{};
    result.underlyingValue_ = underlyingValue_ + other.underlyingValue_;
    return result;
}

constexpr Fixed operator-(Fixed other) const
{
    Fixed result{};
    result.underlyingValue_ = underlyingValue_ - other.underlyingValue_;
    return result;
}
```

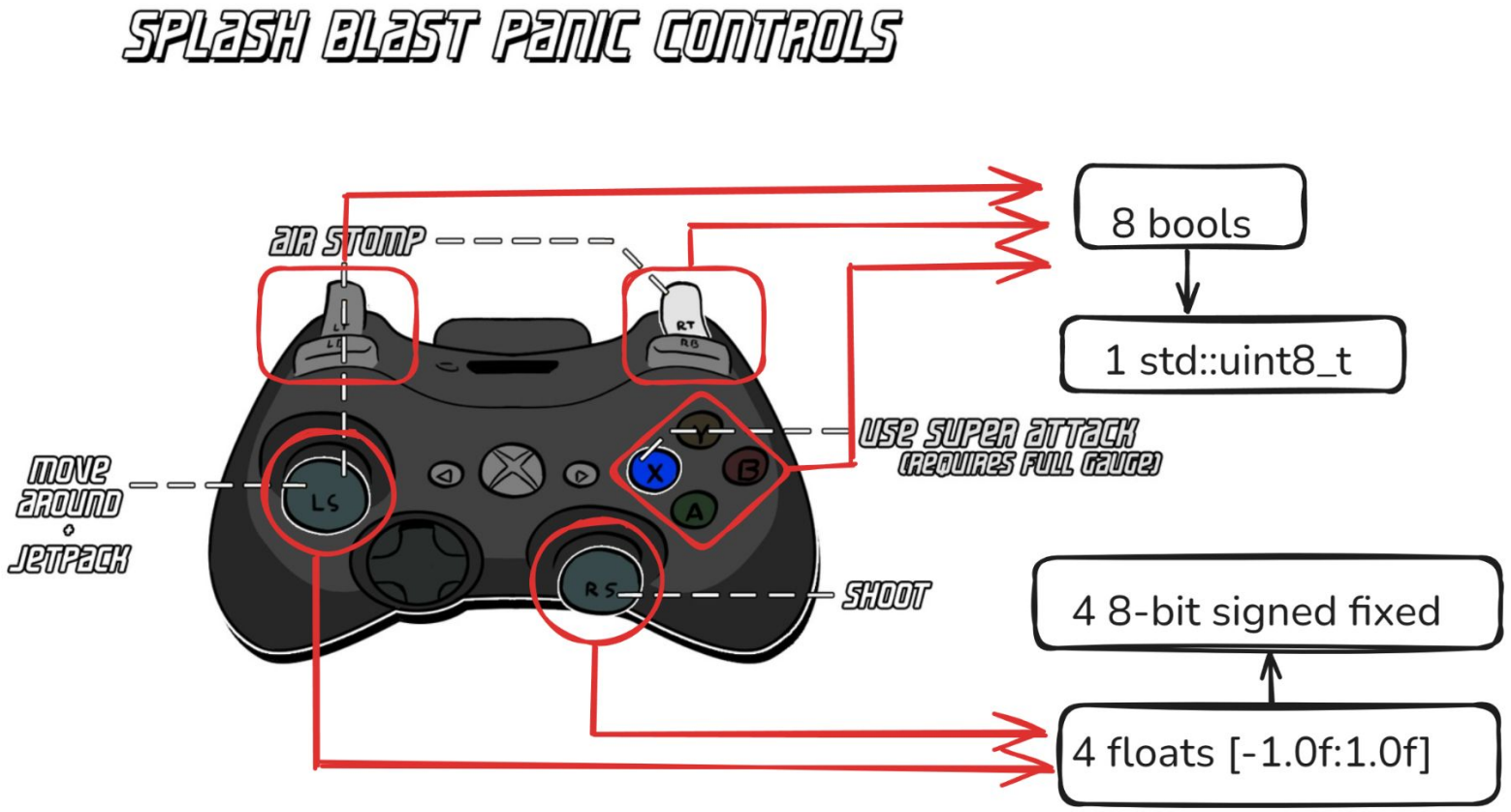
```cpp
constexpr Fixed operator*(Fixed other) const
{
    Fixed result;
    std::int64_t leftOp = static_cast<std::int64_t>(underlyingValue_);
    std::int64_t rightOp = static_cast<std::int64_t>(other.underlyingValue_);
    result.underlyingValue_ = static_cast<std::int32_t>(
      (leftOp * rightOp) >> static_cast<std::int64_t>(Exp));
    return result;
}

constexpr Fixed operator/(Fixed other) const
{
    Fixed result{};
    std::int64_t leftOp = static_cast<std::int64_t>(underlyingValue_) *
      (static_cast<std::int64_t>(1) << static_cast<std::int64_t>(Exp));
    std::int64_t rightOp = static_cast<std::int64_t>(other.underlyingValue_);
    result.underlyingValue_ = static_cast<std::int32_t>(leftOp / rightOp);
    return result;
}
```

SAE INSTITUTE

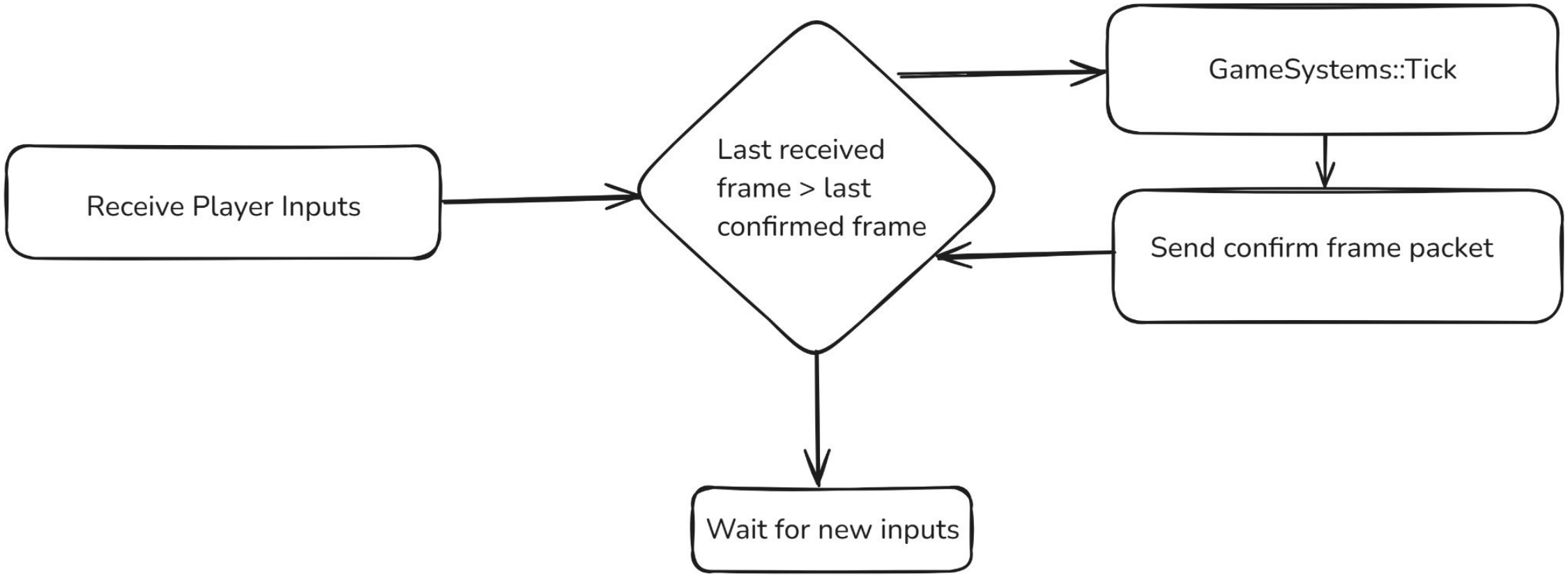
Fixed-point number - LUT

```
int sqrt_table[] = {
0,7326,10360,12689,14652,16381,17945,19383,20721,21978,23167,24298,25378,26415,27412,28374,29305,30206,31
082,31934,32763,33573,34363,35135,35891,36631,37356,38068,38766,39453,40127,40790,41443,42086,42719,43342
,43957,44563,45162,45752,46335,46910,47479,48041,48596,49145,49689,50226,50757,51283,51804,52319,52830,53
335,53836,54332,54824,55311,55795,56274,56748,57219,57686,58150,58610,59066,59518,59967,60413,60856,61295
,61732,62165,62595,63022,63447,63868,64287,64703,65117,65527,65936,66342,66745,67146,67544,67940,68334,68
726,69115,69502,69888,70270,70651,71030,71407,71782,72155,72526,72895,73262,73627,73991,74353,74713,75071
,75428,75783,76136,76488,76838,77186,77533,77879,78222,78565,78906,79245,79583,79919,80255,80588,80921,81
252,81581,81909,82236,82562,82887,83210,83532,83852,84172,84490,84807,85123,85438,85751,86063,86375,86685
,86994,87302,87609,87915,88219,88523,88826,89127,89428,89727,90026,90324,90620,90916,91211,91504,91797,92
089,92380,92670,92959,93247,93535,93821,94107,94392,94676,94959,95241,95522,95803,96082,96361,96639,96917
,97193,97469,97744,98018,98291,98564,98836,99107,99378,99647,99916, ...
```

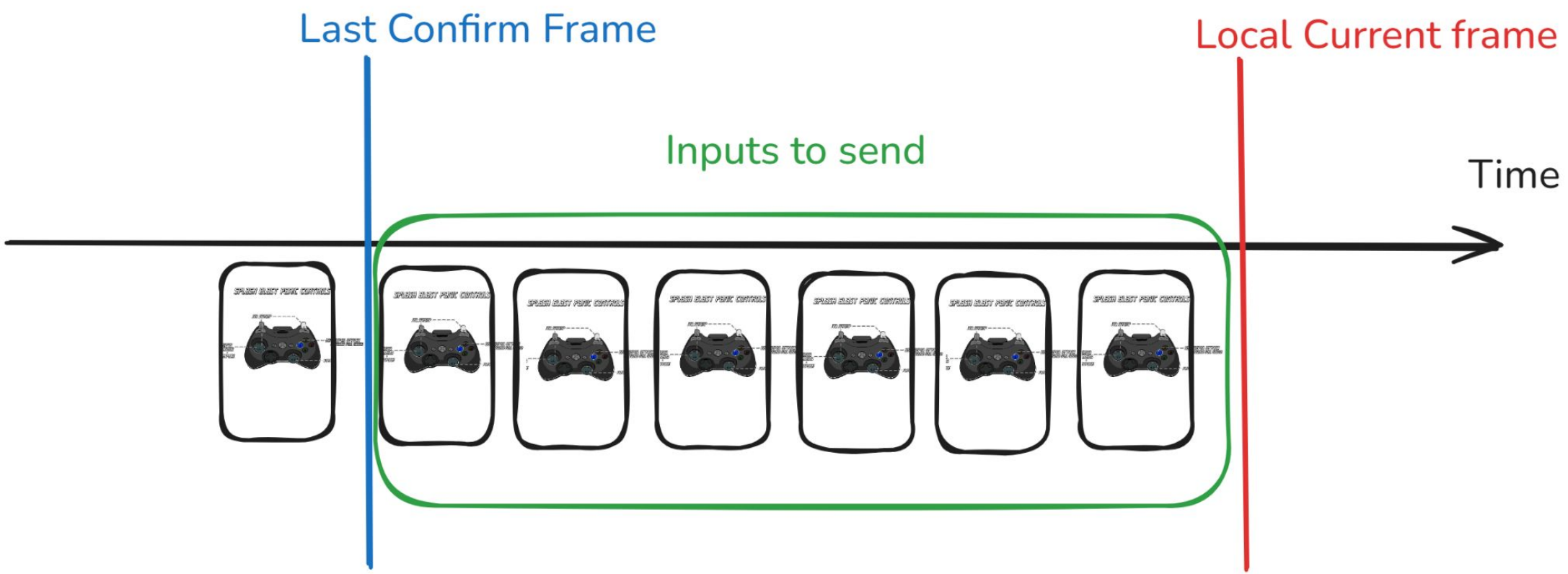
Input Compression



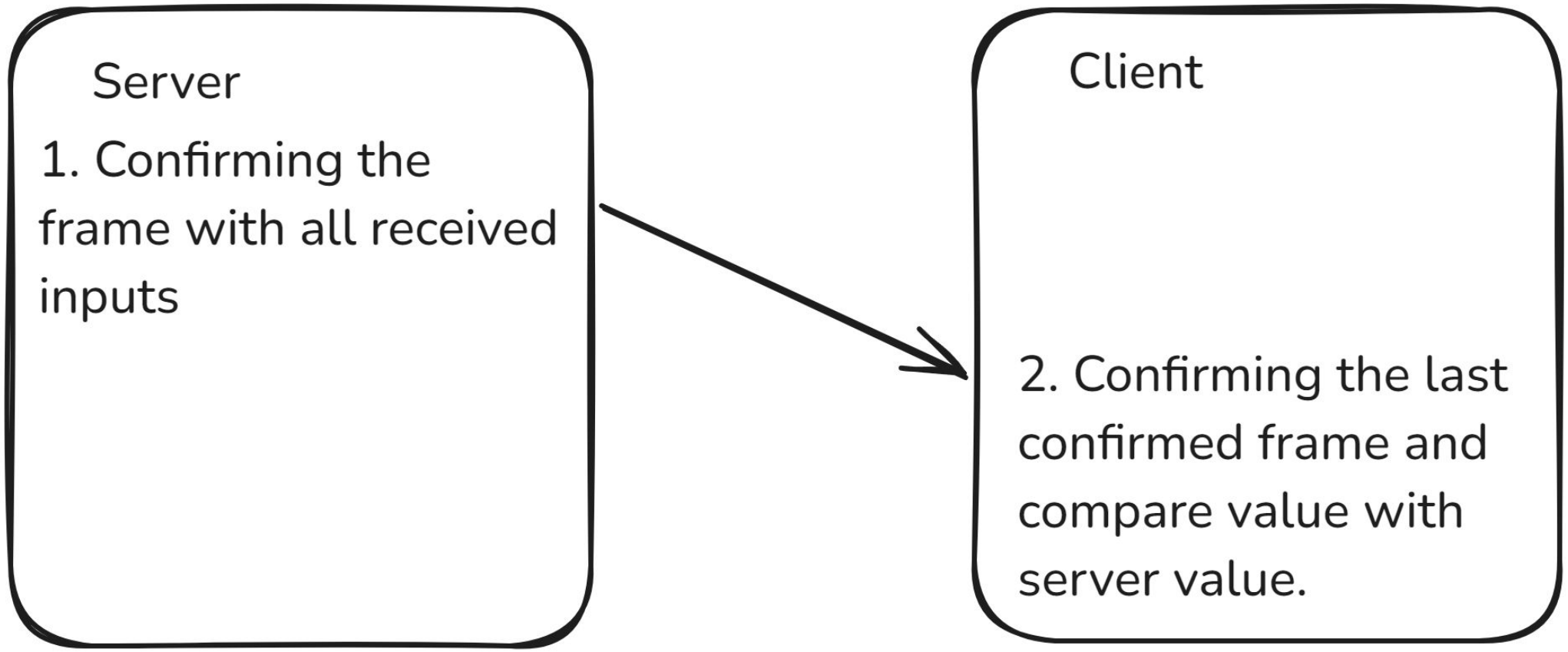
Server/master tick



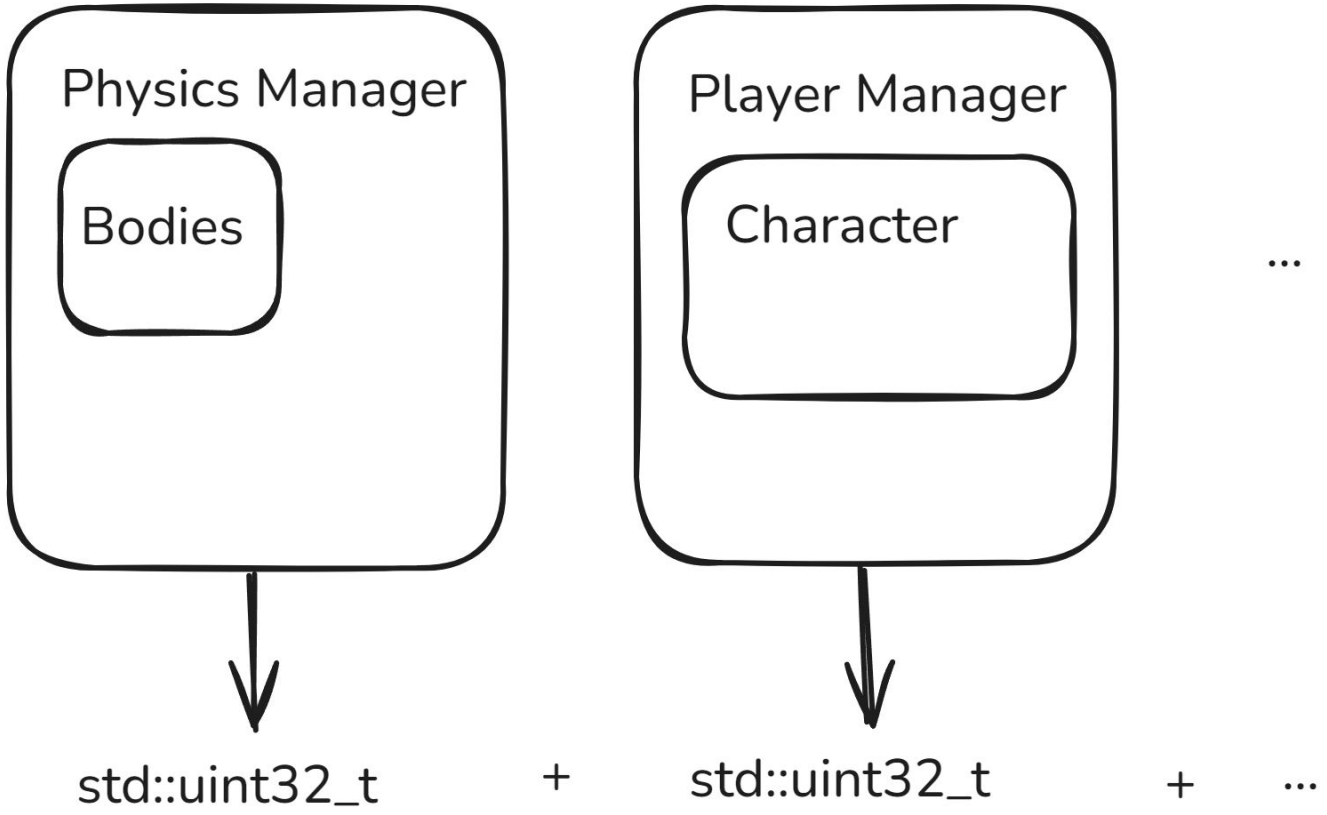
Receiving Player Inputs



Confirm frame



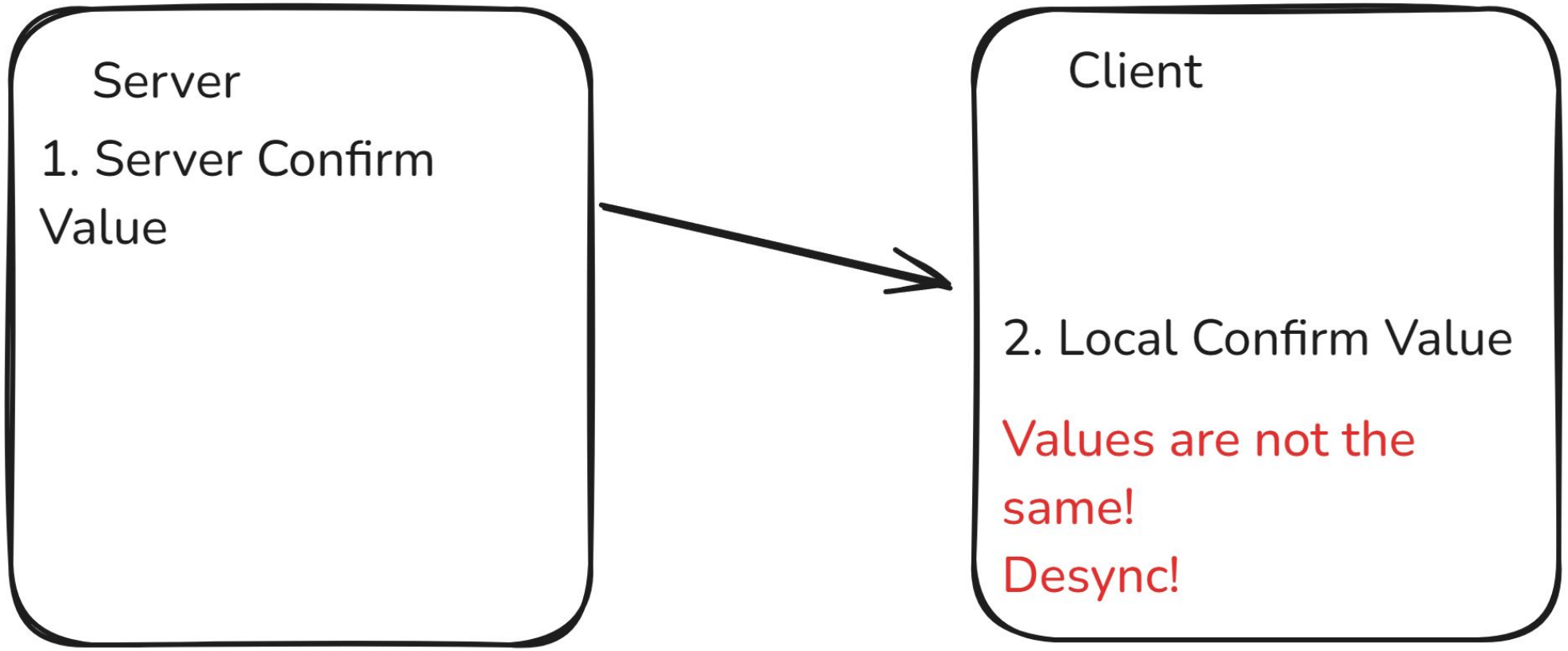
Checksum


Physics Manager
Bodies
Player Manager
Character
...
std::uint32_t
+
std::uint32_t
+
...

Checksum

```
for(int playerNumber = 0; playerNumber < MaxPlayerNmb; playerNumber++)
{
    if(!IsValid(playerNumber))
    {
        continue;
    }
    const auto* player =
        reinterpret_cast<const std::uint32_t*>(&playerCharacters_[playerNumber]);

    for(std::size_t i = 0; i < sizeof(PlayerCharacter)/sizeof(std::uint32_t); i++)
    {
        result += player[i];
    }
}
```

Desync


Server
1. Server Confirm Value
Client
2. Local Confirm Value
Values are not the same!
Desync!

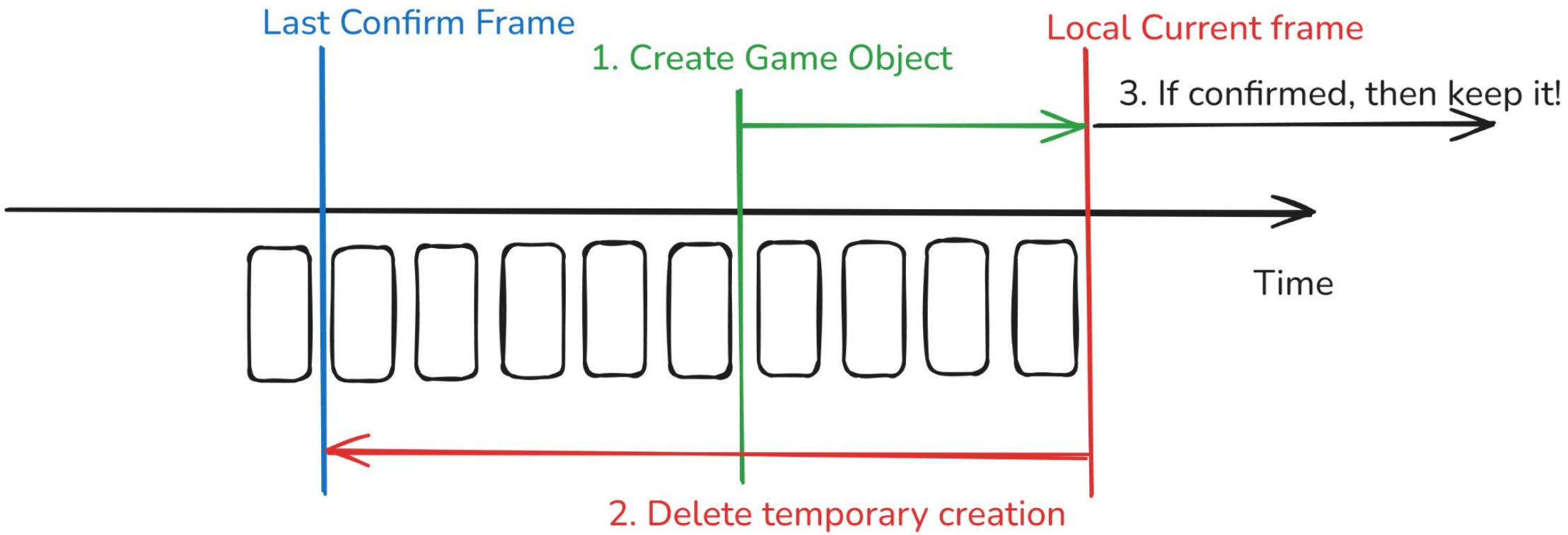
Checksum implementation

```cpp
template<int argCount>
struct Checksum
{
    Checksum(std::initializer_list<std::uint32_t> list )
    {
        for(std::size_t i = 0; i < list.size(); i++)
        {
            data_[i] = data(list)[i];
        }
    }
    bool operator==(const Checksum& other) const;
    bool operator!=(const Checksum& other) const;
    std::uint32_t operator[](std::size_t index) const;

    [[nodiscard]] explicit operator std::uint32_t() const
    {
        return std::accumulate(data_.begin(), data_.end(), 0);
    }
private:
    std::array<std::uint32_t, argCount> data_{};
};
```

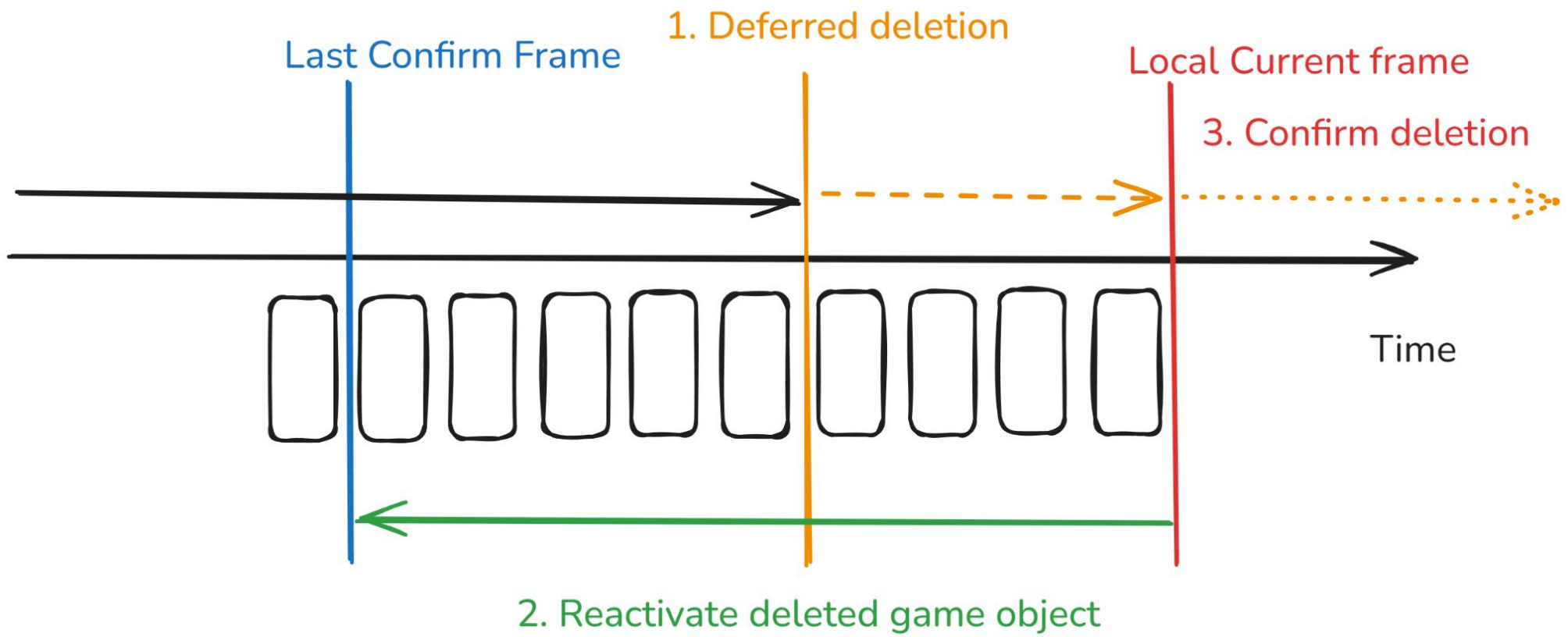
```cpp
enum class ChecksumCategory
{
  PHYSICS,
  PLAYER_MANAGER,
  BULLET_MANAGER,
  LENGTH
};

Checksum<(int)ChecksumCategory::LENGTH>
```

SAE INSTITUTE

Create game objects

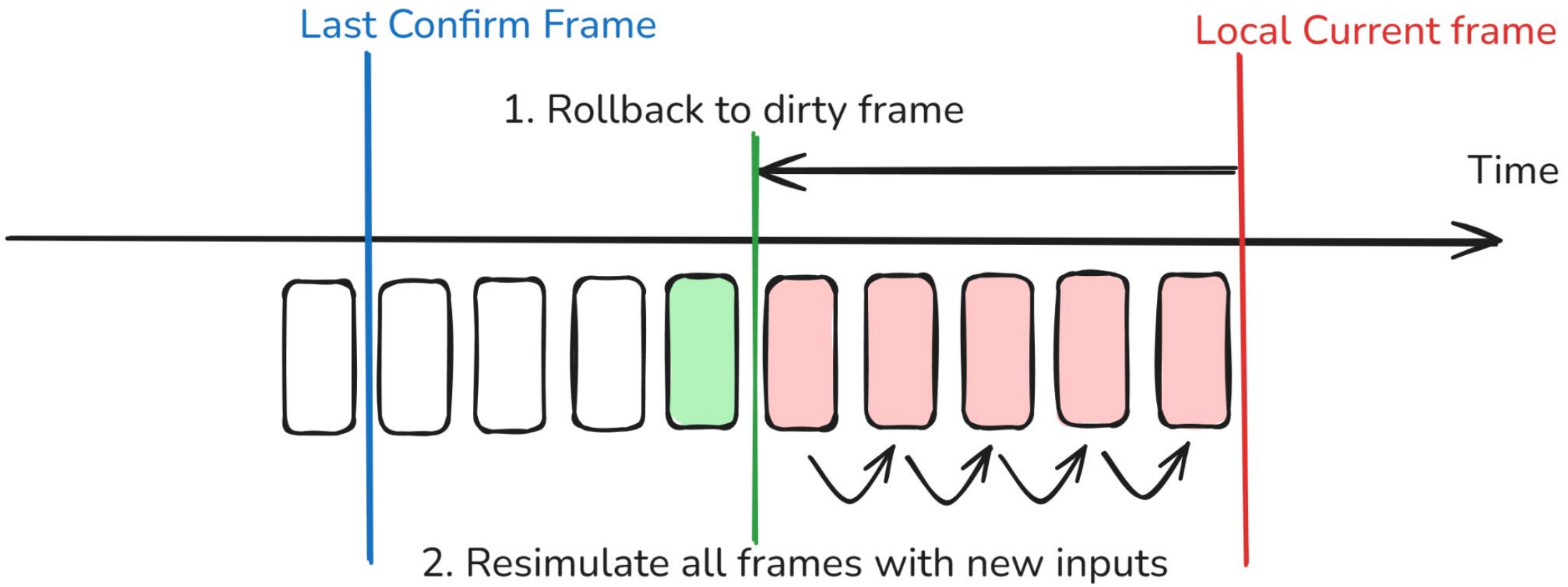


Delete game objects

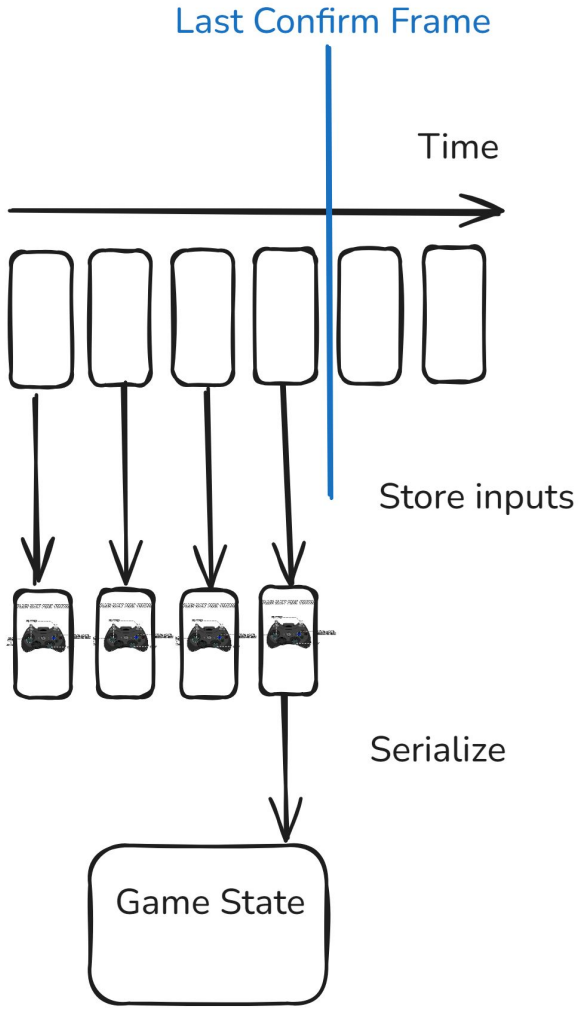


Improvements

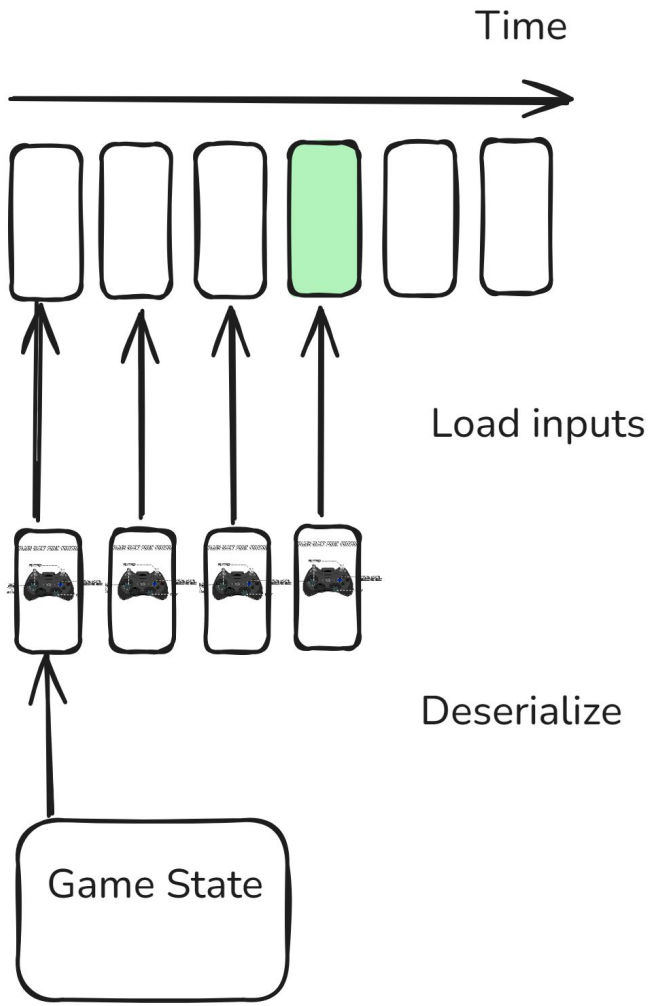
Rollback to dirty frame



Storing game state



Replay



Socket not yet in the standard…

Conclusion

If you want a successful fast-paced online multiplayer experience:

- Deterministic simulation
- Rollback with a simple software architecture
- Debugging desync with checksum

References

- 8 Frames in 16ms: Rollback Networking in Mortal Kombat and Injustice 2 by Netherrealm Games' Michael Stallone, GDC 2018: https://www.youtube.com/watch?v=7jb0FOcImdg
- It IS Rocket Science! The Physics of Rocket League Detailed, by Psyonix's Jared Cone, GDC 2018: https://www.youtube.com/watch?v=ueEmiDM94IE
- Overwatch Gameplay Architecture and Netcode by Blizzard's Timothy Ford, GDC 2017: https://www.youtube.com/watch?v=W3aieHjyNvw
- Fix Your Timestep! By Glenn Fiedler https://gafferongames.com/post/fix_your_timestep/
- 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond, by Paul Bettner: https://www.gamedeveloper.com/programming/1500-archers-on-a-28-8-network-programming-in-age-of-empires-and-beyond
- Games Programming Patterns, Robert Nystrom, https://gameprogrammingpatterns.com/

Online repository

https://github.com/EliasFarhan/SplashOnline

Thank you

You can find the slides here:
https://eliasfarhan.ch/CppCon2024

Follow me on Twitter: @kraklegrand
Or Linkedin: eliasfarhan

Available for consulting or teaching:
elias.farhan@team-kwakwa.com

SAE INSTITUTE