



Loading Code And You

A brief introduction to the loading code in
*Ratchet & Clank® Future: Tools Of
Destruction™*

Giacomino Veltri

10/12/2007

Introduction



- Goals
- Overview
- Problems
- The Future
- Related Topics
- Questions

Goals

- Write cleaner loading code
- Other people can easily ramp up
- Have one main game loop
 - Avoid tight rendering loops
 - Avoid sprinkling render calls throughout code
- Organize different game states
 - Gameplay
 - Movie Player
 - Pause Menu

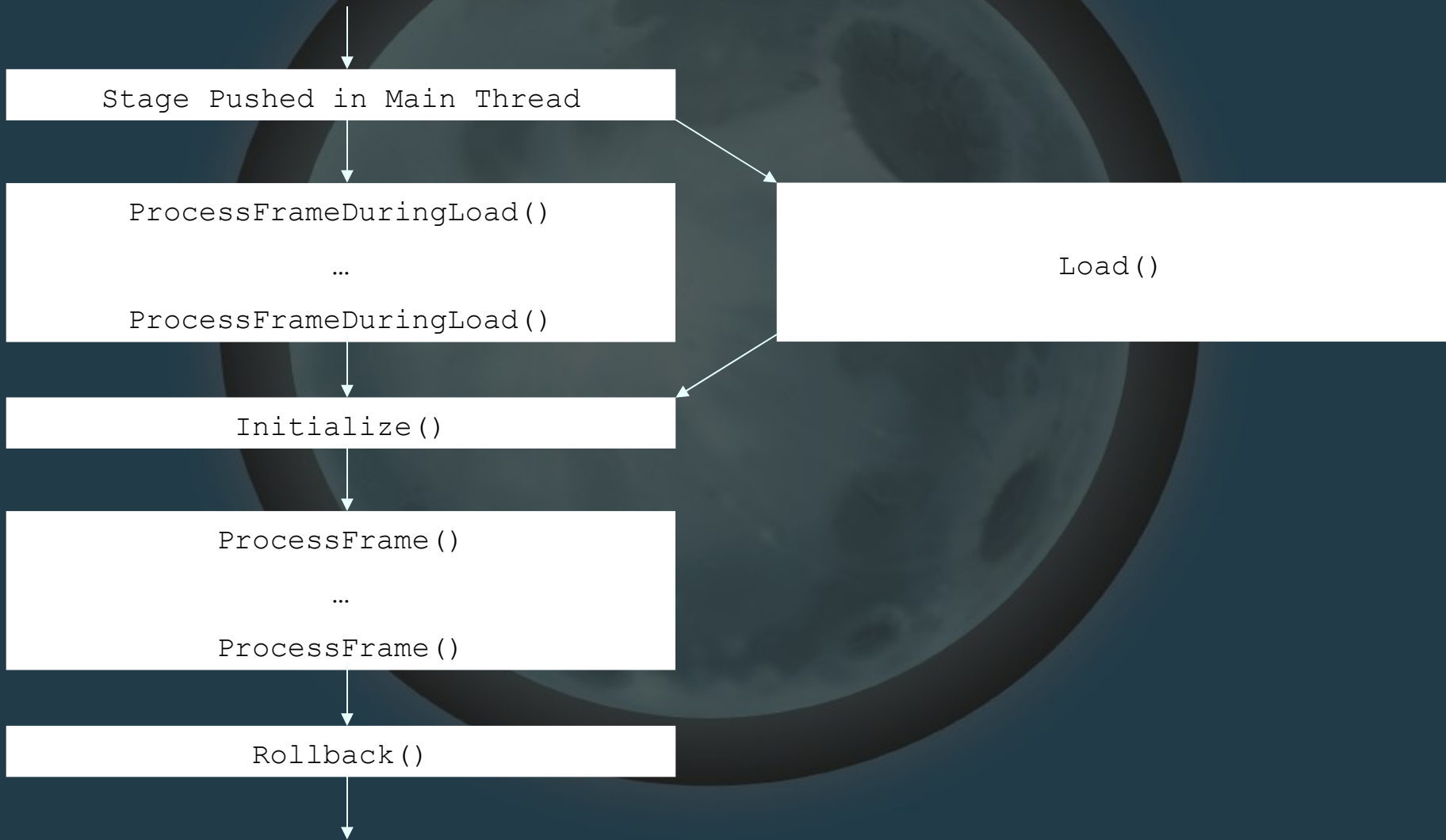
Overview

- A *stage* is an object that loads its data, has its own render loop, and unloads its data.
- A *tier* is a stack of *stages*
 - Transitioning from one tier to the next means popping off stages until a common stage stack is reached and then pushing the unique destination tier's stages.
- A *stage* becomes active when it is pushed onto the current tier stage stack.

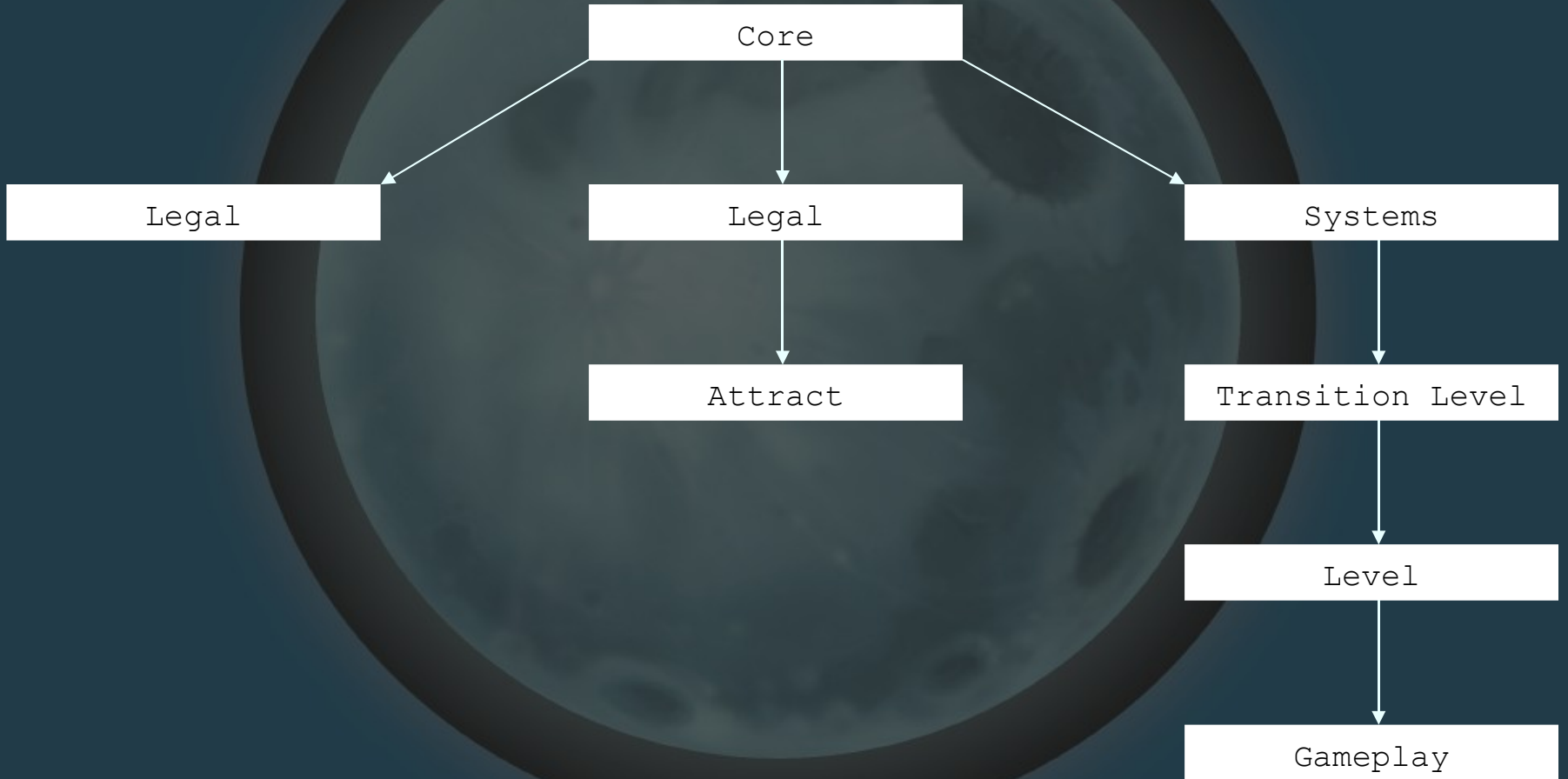
Overview

- The base *stage* class has five virtual functions
 - Load()
 - Initialize()
 - Rollback()
 - ProcessFrame()
 - ProcessFrameDuringLoad()

Overview



Overview



Overview

- Example of a Load() function

```
void Load( void )
{
    ReadChunkFile( WadLoader::g_LevelCachedWad->m_Fx.m_Conduit, global_allocator,
    ...
    CHECK_CANCEL_HORRIBLE_MACRO();
    ReadChunkFile( WadLoader::g_GlobalCachedWad->m_SoundConfig, global_allocator,
    ...
    CHECK_CANCEL_HORRIBLE_MACRO();
    ReadIgFile( WadLoader::g_LevelCachedWad->m_Main, global_allocator, ...
    CHECK_CANCEL_HORRIBLE_MACRO();
    ...
}
```

```
void ReadChunkFile( ... )
{
    ReadFile( ... );
    ParseChunkFile( ... );
}
```

```
void ReadIgFile( ... )
{
    ReadFile( ... );
    ParseIgFile( ... );
}
```

Overview

- Example of a Load() function

```
void ReadFile( ... )
{
    // Wait until movie buffer is sufficiently full (if playing movie)
    // Allocate Memory
    while ( amount_read < file_size )
    {
        bool load_data = MovieBufferPercentFull() > 75 ? true : false;
        if ( load_data )
        {
            [File I/O Library]_scheduler->readFileSync( filename, AMOUNT_TO_READ,
            buffer );
            amount_read += AMOUNT_TO_READ;
        }
        else
        {
            KickOffMovieBufferLoad();
        }
    }
}
```

Overview

- Example of an Initialize() function

```
void Initialize( void )
{
    if ( IG::FindChunkHeader( m_ConduitChunk, chunk, chunk_id ) )
    {
        EffectsConduitData* data = m_ConduitChunk.m_file_header + chunk->m_offset;
        ParseData( data, global_allocator );
    }

    if ( IG::FindChunkHeader( m_SoundConfigChunk, chunk, chunk_id ) )
    {
        gpSoundConfigElems* data = m_SoundConfigChunk.m_file_header +
        chunk>m_offset;
        SoundConfigInit(data);
    }

    ...
}
```

Problems (Fixed)

- ReadFile() allocates memory
 - Can cause bizarre memory stomping
- ReadFile() Movie calls not thread safe
 - Small window of time where accessing a movie handle in a second thread is invalid
 - Problematic mostly for multiple movie playing
- Could not convert all modes to tiers
 - Pause Menu
 - In-Game Movies

Problems (Fixed)

- Cancelling a Load() is not well-supported
 - Added the equivalent of LOAD_FRAME() macro to check for cancel after each function
 - Made quitting the game difficult at times (TRC violations)
 - Main thread stuck waiting for the load thread
 - Load thread does not properly clean up
 - Fixed but not ideal.

Problems (Fixed)

- Movie stuttered during Load
 - Timing difficulties because there are many threads
 - Main Thread
 - Load Thread
 - [File I/O Library] IO Thread
 - [Movie Player] IO Thread
 - Used SPU decompression to alleviate load pressure

The Future

- Move Load() to the main thread
 - Asynchronous load calls instead of synchronous load calls in async thread.
 - Better support for cancelling loads
- Add Setup() function to stages
 - Pre-load initialization like allocate memory
- Handle load errors better
 - Disc eject and quitting
- Make all game-modes tiers

Related Topics



- [File I/O Library]
- Wad System
- Disc Building and Layout

Questions

