Specializing Godot

Panagiotis Tsiapkolis (Panthavma)

October 11th, 2024

What engine to use?

General Purpose Engines (GP)

Flexible, but the workflow can be a bit inefficient

Specialized Engines

Efficient but quickly limited

Custom Engine

Can be optimal, but long and hard to make

Is there a better option?

What engine to use?

General Purpose Engines (GP)

Flexible, but the workflow can be a bit inefficient

Specialized Engines

Efficient but quickly limited

Custom Engine

Can be optimal, but long and hard to make

Is there a better option?

What engine to use?

General Purpose Engines (GP)

Flexible, but the workflow can be a bit inefficient

Specialized Engines

Efficient but quickly limited

Custom Engine

Can be optimal, but long and hard to make

Is there a better option?

What engine to use?

General Purpose Engines (GP)

Flexible, but the workflow can be a bit inefficient

Specialized Engines

Efficient but quickly limited

Custom Engine

Can be optimal, but long and hard to make

Is there a better option?

What engine to use?

General Purpose Engines (GP)

Flexible, but the workflow can be a bit inefficient

Specialized Engines

Efficient but quickly limited

Custom Engine

Can be optimal, but long and hard to make

Is there a better option?

Can we specialize a General Purpose Engine for our workflow?

- **Why?** What advantages can we get?
- **How?** How to integrate it in productions?
- **Users**: How to enable them?

Example: Castagne, Fighting Game creation framework

Can we specialize a General Purpose Engine for our workflow?

- **Why?** What advantages can we get?
- **How?** How to integrate it in productions?
- **Users**: How to enable them?

Example: Castagne, Fighting Game creation framework

Can we specialize a General Purpose Engine for our workflow?

- **Why?** What advantages can we get?
- **How?** How to integrate it in productions?
- **Users**: How to enable them?

Example: Castagne, Fighting Game creation framework

Can we specialize a General Purpose Engine for our workflow?

- **Why?** What advantages can we get?
- **How?** How to integrate it in productions?
- **Users**: How to enable them?

Example: Castagne, Fighting Game creation framework

Can we specialize a General Purpose Engine for our workflow?

- **Why?** What advantages can we get?
- **How?** How to integrate it in productions?
- **Users**: How to enable them?

Example: Castagne, Fighting Game creation framework

Who am I?

3D Programmer / Rendering Engineer

Currently finishing up a PhD on stylized rendering for games!

Mostly self-taught

Been using Godot for 7 years now, and programming for 20 now.

My specialties

I like software architecture and rendering! Math too.

Who am I?

3D Programmer / Rendering Engineer

Currently finishing up a PhD on stylized rendering for games!

Mostly self-taught

Been using Godot for 7 years now, and programming for 20 now.

My specialties

I like software architecture and rendering! Math too.

Who am I?

3D Programmer / Rendering Engineer

Currently finishing up a PhD on stylized rendering for games!

Mostly self-taught

Been using Godot for 7 years now, and programming for 20 now.

My specialties

I like software architecture and rendering! Math too.

Who am I?

3D Programmer / Rendering Engineer

Currently finishing up a PhD on stylized rendering for games!

Mostly self-taught

Been using Godot for 7 years now, and programming for 20 now.

My specialties

I like software architecture and rendering! Math too.


Hey you! Did you hear about
Castagne!
Fighting Game Engine?
Free and Open Source based on Godot!
The superb Castagne Editor!
Rollback netcode!
Very flexible! Can handle many genres!
2D Fighters! 3D Fighters! Arena Fighter!
Platform Fighter! Beat them up! Character Action!
Get it on
castagneengine.com !

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

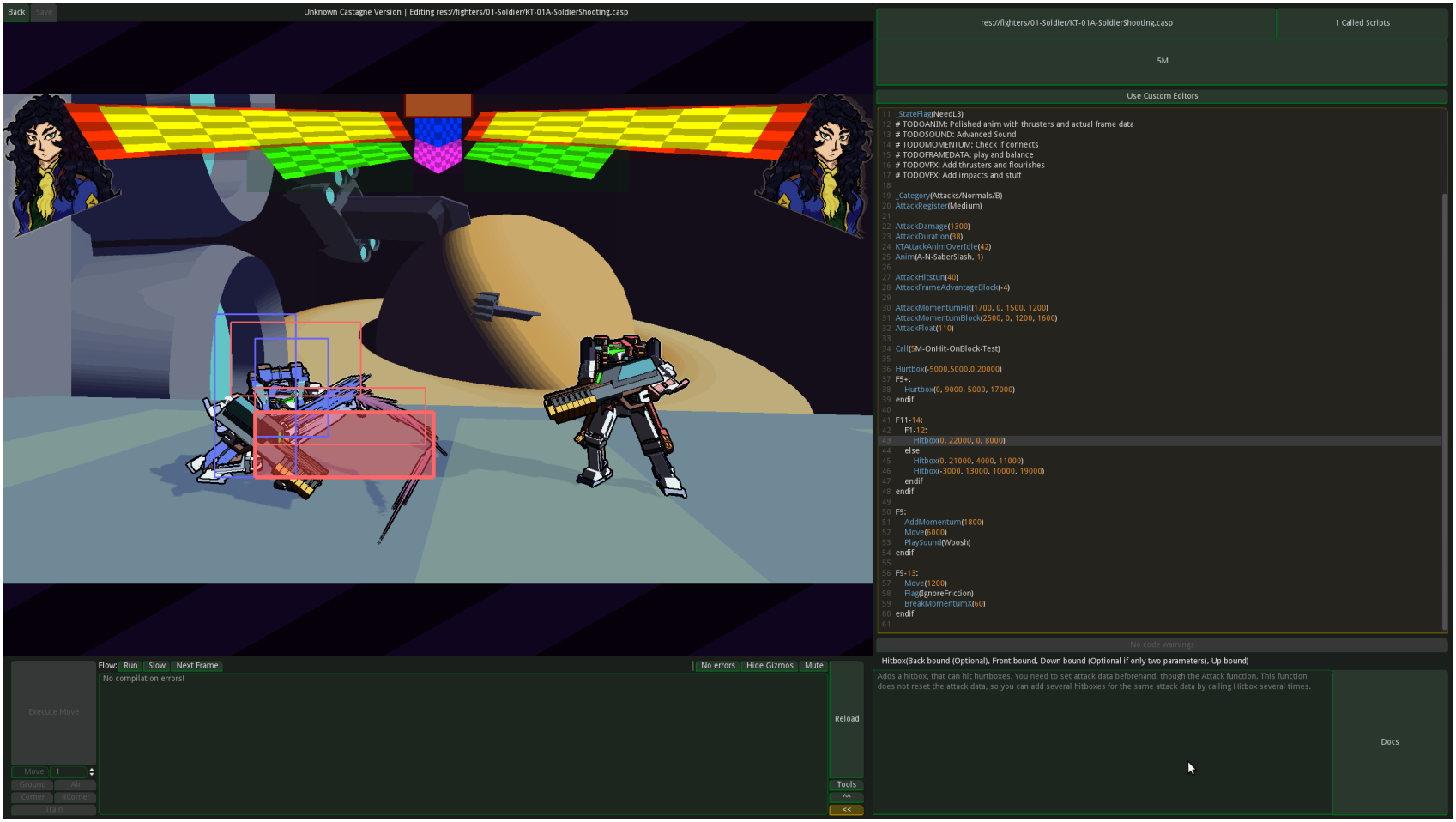
- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features

Castagne

Castagne's Objective

Make fighting / action games efficiently

- Editor
- Custom Language
- Extensible
- Rollback Netcode
- Physics Engine
- Collaboration Features


Back
Save
Unknown Castagne Version | Editing res://fighters/01-Soldier/KT-01A-SoldierShooting.casp
res://fighters/01-Soldier/KT-01A-SoldierShooting.casp
1 Called Scripts
5M
Use Custom Editors
_StateFlag(NeedL3)
# TODOANIM: Polished anim with thrusters and actual frame data
# TODOSOUND: Advanced Sound
# TODOMOMENTUM: Check if connects
# TODOFRAMEDATA: play and balance
# TODOVFX: Add thrusters and flourishes
# TODOVFX: Add impacts and stuff
_Category(Attacks/Normals/B)
AttackRegister(Medium)
AttackDamage(1300)
AttackDuration(38)
KTAttackAnimOverIdle(42)
Anim(A-N-SaberSlash, 1)
AttackHitstun(40)
AttackFrameAdvantageBlock(-4)
AttackMomentumHit(1700, 0, 1500, 1200)
AttackMomentumBlock(2500, 0, 1200, 1600)
AttackFloat(110)
Call(5M-OnHit-OnBlock-Test)
Hurtbox(-5000,5000,0,20000)
F5+:
Hurtbox(0, 9000, 5000, 17000)
endif
F11-14:
F1-12:
Hitbox(0, 22000, 0, 8000)
else
Hitbox(0, 21000, 4000, 11000)
Hitbox(-3000, 13000, 10000, 19000)
endif
endif
F9:
AddMomentum(1800)
Move(6000)
PlaySound(Woosh)
endif
F9-13:
Move(1200)
Flag(IgnoreFriction)
BreakMomentumX(60)
endif
No code warnings
Hitbox(Back bound (Optional), Front bound, Down bound (Optional if only two parameters), Up bound)
Adds a hitbox, that can hit hurtboxes. You need to set attack data beforehand, though the Attack function. This function does not reset the attack data, so you can add several hitboxes for the same attack data by calling Hitbox several times.
Docs
Flow:
Run
Slow
Next Frame
No errors
Hide Gizmos
Mute
No compilation errors!
Execute Move
Move
1
Ground
Air
Corner
RCorner
Train
Reload
Tools
^^
<<

Why do it? Unique Needs

- **Performance?** Might need unique optimizations
- **Complex?** Might need specialized tools
- **Balancing?** Too much for an indie team
- **Ambitious?** You may need stellar tools

Does your game have a unique constraint?

Castagne needs to implement *Rollback Netcode*, which drives a lot of the technical decisions

Why do it? Unique Needs

- **Performance?** Might need unique optimizations
- **Complex?** Might need specialized tools
- **Balancing?** Too much for an indie team
- **Ambitious?** You may need stellar tools

Does your game have a unique constraint?

Castagne needs to implement *Rollback Netcode*, which drives a lot of the technical decisions

Why do it? Unique Needs

- **Performance?** Might need unique optimizations
- **Complex?** Might need specialized tools
- **Balancing?** Too much for an indie team
- **Ambitious?** You may need stellar tools

Does your game have a unique constraint?

Castagne needs to implement *Rollback Netcode*, which drives a lot of the technical decisions

Why do it? Unique Needs

- **Performance?** Might need unique optimizations
- **Complex?** Might need specialized tools
- **Balancing?** Too much for an indie team
- **Ambitious?** You may need stellar tools

Does your game have a unique constraint?

Castagne needs to implement *Rollback Netcode*, which drives a lot of the technical decisions

Why do it? Unique Needs

- **Performance?** Might need unique optimizations
- **Complex?** Might need specialized tools
- **Balancing?** Too much for an indie team
- **Ambitious?** You may need stellar tools

Does your game have a unique constraint?

Castagne needs to implement *Rollback Netcode*, which drives a lot of the technical decisions

The key: Iteration Time

- **Make more game**: The challenge is to *"fill the disk"*.
- **Momentum**: Can prototype and try it out quickly.
- **Quality**: More iterations allow more polish.

Fast iteration is transformative

You don't only get more game and more quality, you also try out new ideas you wouldn't otherwise

The key: Iteration Time

- **Make more game**: The challenge is to *"fill the disk"*.
- **Momentum**: Can prototype and try it out quickly.
- **Quality**: More iterations allow more polish.

Fast iteration is transformative

You don't only get more game and more quality, you also try out new ideas you wouldn't otherwise

The key: Iteration Time

- **Make more game**: The challenge is to *"fill the disk"*.
- **Momentum**: Can prototype and try it out quickly.
- **Quality**: More iterations allow more polish.

Fast iteration is transformative

You don't only get more game and more quality, you also try out new ideas you wouldn't otherwise

The key: Iteration Time

- **Make more game**: The challenge is to *"fill the disk"*.
- **Momentum**: Can prototype and try it out quickly.
- **Quality**: More iterations allow more polish.

Fast iteration is transformative

You don't only get more game and more quality, you also try out new ideas you wouldn't otherwise


Return
Save
Decide
EQUIPMENT
Iron Boots
500


Return
Save
Decide
EQUIPMENT
Iron Boots
500


Return
Save
Decide
EQUIPMENT
Iron Boots
500
Water Temple
200


Return
Save
Decide
Iron Boots
500
200


Return
Save
Decide
Iron Boots
500
Water T
200
Return
Save
Decide
Iron Boots
500


Return
Save
Decide
Iron Boots
500
200


Return
Save
Decide
Iron Boots
500
200


Return
Save
Decide
Iron Boots
500
Water Temple
200


Return
Save
Decide
Iron Boots
500
Water Temple
200


Return
Save
Decide
Iron Boots
500
Water Temple
200

Domain specific tools

Your advantage: you know your domain

You can create specialized tools that only have what you *need*

- You also know what you *don't need*: make a focused tool
- Can create tools for any core system: bullet path creation in Danmakus, path visualization / navigation in platformers...
- Domain Specific Languages (DSL) can be excellent for complex gameplay behavior

Castagne's tools

CASP, a DSL for making fighting games attacks and moves, and a strong editor with state manipulation to test them in various scenarios (air hit, corner...)

Domain specific tools

Your advantage: you know your domain

You can create specialized tools that only have what you *need*

- You also know what you *don't need*: make a focused tool
- Can create tools for any core system: bullet path creation in Danmakus, path visualization / navigation in platformers...
- Domain Specific Languages (DSL) can be excellent for complex gameplay behavior

Castagne's tools

CASP, a DSL for making fighting games attacks and moves, and a strong editor with state manipulation to test them in various scenarios (air hit, corner...)

Domain specific tools

Your advantage: you know your domain

You can create specialized tools that only have what you *need*

- You also know what you *don't need*: make a focused tool
- Can create tools for any core system: bullet path creation in Danmakus, path visualization / navigation in platformers...
- Domain Specific Languages (DSL) can be excellent for complex gameplay behavior

Castagne's tools

CASP, a DSL for making fighting games attacks and moves, and a strong editor with state manipulation to test them in various scenarios (air hit, corner...)

Domain specific tools

Your advantage: you know your domain

You can create specialized tools that only have what you *need*

- You also know what you *don't need*: make a focused tool
- Can create tools for any core system: bullet path creation in Danmakus, path visualization / navigation in platformers...
- Domain Specific Languages (DSL) can be excellent for complex gameplay behavior

Castagne's tools

CASP, a DSL for making fighting games attacks and moves, and a strong editor with state manipulation to test them in various scenarios (air hit, corner...)

Domain specific tools

Your advantage: you know your domain

You can create specialized tools that only have what you *need*

- You also know what you *don't need*: make a focused tool
- Can create tools for any core system: bullet path creation in Danmakus, path visualization / navigation in platformers...
- Domain Specific Languages (DSL) can be excellent for complex gameplay behavior

Castagne's tools

CASP, a DSL for making fighting games attacks and moves, and a strong editor with state manipulation to test them in various scenarios (air hit, corner...)

Gameplay Tests

- You can implement **in-depth tests** for your gameplay
- *Can the character jump 3-block diagonal gaps?*
- *What ressources and units does this build order get me?*
- *What's the time to kill at various ranges and accuracies?*

Castagne's Combo Unit Tests

Replay an input, does it still combo? How much damage?

Gameplay Tests

- You can implement **in-depth tests** for your gameplay
- *Can the character jump 3-block diagonal gaps?*
- *What ressources and units does this build order get me?*
- *What's the time to kill at various ranges and accuracies?*

Castagne's Combo Unit Tests

Replay an input, does it still combo? How much damage?

Gameplay Tests

- You can implement **in-depth tests** for your gameplay
- *Can the character jump 3-block diagonal gaps?*
- *What ressources and units does this build order get me?*
- *What's the time to kill at various ranges and accuracies?*

Castagne's Combo Unit Tests

Replay an input, does it still combo? How much damage?

Gameplay Tests

- You can implement **in-depth tests** for your gameplay
- *Can the character jump 3-block diagonal gaps?*
- *What ressources and units does this build order get me?*
- *What's the time to kill at various ranges and accuracies?*

Castagne's Combo Unit Tests

Replay an input, does it still combo? How much damage?

Gameplay Tests

- You can implement **in-depth tests** for your gameplay
- *Can the character jump 3-block diagonal gaps?*
- *What ressources and units does this build order get me?*
- *What's the time to kill at various ranges and accuracies?*

Castagne's Combo Unit Tests

Replay an input, does it still combo?
How much damage?

Production

It's a complex and massive project!

Good tools take a long time to create, you must be able to reuse them though good *code architecture* and *project management*.

- How to find that reuse potential
- How to not waste time on less impactful code
- How to produce games while the tech is being built

Production

It's a complex and massive project!

Good tools take a long time to create, you must be able to reuse them though good *code architecture* and *project management*.

- How to find that reuse potential
- How to not waste time on less impactful code
- How to produce games while the tech is being built

Production

It's a complex and massive project!

Good tools take a long time to create, you must be able to reuse them though good *code architecture* and *project management*.

- How to find that reuse potential
- How to not waste time on less impactful code
- How to produce games while the tech is being built

Production

It's a complex and massive project!

Good tools take a long time to create, you must be able to reuse them though good *code architecture* and *project management*.

- How to find that reuse potential
- How to not waste time on less impactful code
- How to produce games while the tech is being built

Find the technical core

- What is the **technical core** of your game and your tools: what does the *code* rely on?

Project yourself in the future

- What do I want to make?
- How do I want to work?

- You can imagine "How to do [GAME] in my tools?", with either an existing game or a project that you want to make
- Don't make your scope explode, but keep the doors open!

Find the technical core

- What is the **technical core** of your game and your tools: what does the *code* rely on?

Project yourself in the future

- What do I want to make?
- How do I want to work?

- You can imagine "How to do [GAME] in my tools?", with either an existing game or a project that you want to make
- Don't make your scope explode, but keep the doors open!

Find the technical core

- What is the **technical core** of your game and your tools: what does the *code* rely on?

Project yourself in the future

- What do I want to make?
- How do I want to work?

- You can imagine "How to do [GAME] in my tools?", with either an existing game or a project that you want to make
- Don't make your scope explode, but keep the doors open!

Find the technical core

- What is the **technical core** of your game and your tools: what does the *code* rely on?

Project yourself in the future

- What do I want to make?
- How do I want to work?

- You can imagine "How to do [GAME] in my tools?", with either an existing game or a project that you want to make
- Don't make your scope explode, but keep the doors open!

Find the technical core

- What is the **technical core** of your game and your tools: what does the *code* rely on?

Project yourself in the future

- What do I want to make?
- How do I want to work?

- You can imagine "How to do [GAME] in my tools?", with either an existing game or a project that you want to make
- Don't make your scope explode, but keep the doors open!

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Castagne's Technical Core

Fighting games have a lot of systems in common:

- **Attack Logic**: Startup, recovery
- **Inputs**: Relative to the opponent, motions
- **Cancels**: Normal, special, dash...

Castagne's core: State Machines

Every entity is a complex state machine, and the tools revolve around how to author and ajust it.

What other genres have a similar technical core?

- **Beat them All / Character Action**: Versus multiple AIs
- **Platformers**: Lots of jump types and states

Relying on Godot

- Godot has already a lot of tools, you should use them!

Ideal: Clear separation of concerns

You should rewrite a part only if it gives you tangible benefit

- Even then: can you just change the interface to that part?

Relying on Godot

- Godot has already a lot of tools, you should use them!

Ideal: Clear separation of concerns

You should rewrite a part only if it gives you tangible benefit

- Even then: can you just change the interface to that part?

Relying on Godot

- Godot has already a lot of tools, you should use them!

Ideal: Clear separation of concerns

You should rewrite a part only if it gives you tangible benefit

- Even then: can you just change the interface to that part?

Relying on Godot - Castagne

Separation

Castagne handles core gameplay, Godot handles rendering, UI, Input.

Physics

Needs to be cross platform determinist, no floats. Needs a new physics engine.

Graphics

Only needs to interface with the gameplay. Can reuse all the import procedures, rendering methods, etc.

Relying on Godot - Castagne

Separation

Castagne handles core gameplay, Godot handles rendering, UI, Input.

Physics

Needs to be cross platform determinist, no floats. Needs a new physics engine.

Graphics

Only needs to interface with the gameplay. Can reuse all the import procedures, rendering methods, etc.

Relying on Godot - Castagne

Separation

Castagne handles core gameplay, Godot handles rendering, UI, Input.

Physics

Needs to be cross platform determinist, no floats. Needs a new physics engine.

Graphics

Only needs to interface with the gameplay. Can reuse all the import procedures, rendering methods, etc.

Dev and Prod

- Long and complex tool development must also make a game to ensure quality and suitability (ie Dogfooding)
- Shorter projects help reduce risk and drive the tool faster

Kronian Titans

Ambitious project that drives the need for the advanced tools

Molten Winds Open Edition

Short project which enables fast development and iteration for Castagne

Dev and Prod

- Long and complex tool development must also make a game to ensure quality and suitability (ie Dogfooding)
- Shorter projects help reduce risk and drive the tool faster

Kronian Titans

Ambitious project that drives the need for the advanced tools

Molten Winds Open Edition

Short project which enables fast development and iteration for Castagne

Dev and Prod

- Long and complex tool development must also make a game to ensure quality and suitability (ie Dogfooding)
- Shorter projects help reduce risk and drive the tool faster

Kronian Titans

Ambitious project that drives the need for the advanced tools

Molten Winds Open Edition

Short project which enables fast development and iteration for Castagne

Dev and Prod

- Long and complex tool development must also make a game to ensure quality and suitability (ie Dogfooding)
- Shorter projects help reduce risk and drive the tool faster

Kronian Titans

Ambitious project that drives the need for the advanced tools

Molten Winds Open Edition

Short project which enables fast development and iteration for Castagne

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Roadmap

Direct your roadmap to be able to advance in parallel of your games and it's needs by balancing a few factors (non-exhaustive):

Current Needs

Features, blockers, etc.

Future Needs

Only those you know will happen!

Technical Constraints

Performance, architecture

Logistics

The team and their time

It's not static: revise often!

Castagne's roadmap

GAMEPLAY FIRST

Priority to iterating on your game's core. The rest (graphics, menus...) come after.

Technical soundness is key

Spend extra time if needed. Projects in a hurry can implement a hacky solution.

Stay flexible!

High Impact Features will present themselves: a cool tool, a community need, or a hype idea

Castagne's roadmap

GAMEPLAY FIRST

Priority to iterating on your game's core. The rest (graphics, menus...) come after.

Technical soundness is key

Spend extra time if needed. Projects in a hurry can implement a hacky solution.

Stay flexible!

High Impact Features will present themselves: a cool tool, a community need, or a hype idea

Castagne's roadmap

GAMEPLAY FIRST

Priority to iterating on your game's core. The rest (graphics, menus...) come after.

Technical soundness is key

Spend extra time if needed. Projects in a hurry can implement a hacky solution.

Stay flexible!

High Impact Features will present themselves: a cool tool, a community need, or a hype idea

Experiments

- You can't plan everything, you must allow time for experimentation
- New paths will appear during development, you want to be able to take them
- Good code is code that is iterated upon.
- Use your tech debt as a ressource and plan around it

Castagne's CASP Language

Not in the initial draft, evolved a lot through discovery

Experiments

- You can't plan everything, you must allow time for experimentation
- New paths will appear during development, you want to be able to take them
- Good code is code that is iterated upon.
- Use your tech debt as a ressource and plan around it

Castagne's CASP Language

Not in the initial draft, evolved a lot through discovery

Experiments

- You can't plan everything, you must allow time for experimentation
- New paths will appear during development, you want to be able to take them
- Good code is code that is iterated upon.
- Use your tech debt as a ressource and plan around it

Castagne's CASP Language

Not in the initial draft, evolved a lot through discovery

Experiments

- You can't plan everything, you must allow time for experimentation
- New paths will appear during development, you want to be able to take them
- Good code is code that is iterated upon.
- Use your tech debt as a ressource and plan around it

Castagne's CASP Language

Not in the initial draft, evolved a lot through discovery

Experiments

- You can't plan everything, you must allow time for experimentation
- New paths will appear during development, you want to be able to take them
- Good code is code that is iterated upon.
- Use your tech debt as a ressource and plan around it

Castagne's CASP Language

Not in the initial draft, evolved a lot through discovery

Don't break production...

- ...too much. Be mindful.
- Still, in the early phase keep in mind that it should be an option, as you figure out how it should work.
- Tools exist: RegEx, or one time python scripts

Cool SCM Tip

You can use a *git submodule* for your specialization and branches to advance in parallel!

Don't break production...

- ...too much. Be mindful.
- Still, in the early phase keep in mind that it should be an option, as you figure out how it should work.
- Tools exist: RegEx, or one time python scripts

Cool SCM Tip

You can use a *git submodule* for your specialization and branches to advance in parallel!

Don't break production...

- ...too much. Be mindful.
- Still, in the early phase keep in mind that it should be an option, as you figure out how it should work.
- Tools exist: RegEx, or one time python scripts

Cool SCM Tip

You can use a *git submodule* for your specialization and branches to advance in parallel!

Don't break production...

- ...too much. Be mindful.
- Still, in the early phase keep in mind that it should be an option, as you figure out how it should work.
- Tools exist: RegEx, or one time python scripts

Cool SCM Tip

You can use a *git submodule* for your specialization and branches to advance in parallel!

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

Who are your users?

- *Who is using your tool?*
- *What is their knowledge / skills?*
- *What do they want to achieve?*
- *What are their main / daily tasks?*

Define your users!

Don't just create a tool to make that game, create the *the tool that fits your team / users*.

- Castagne's target user: me (and similar experts)
- Secondary user: beginners, with low coding experience

How does your team interact?

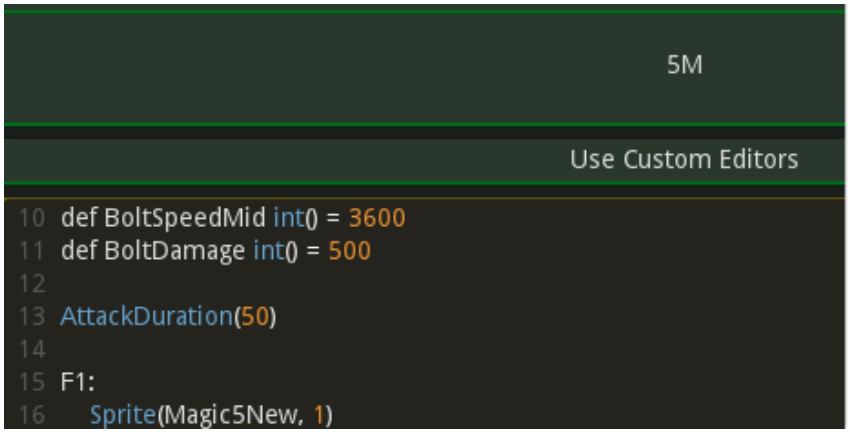
- What jobs need to be done before others?
- How do YOU want your team to interact?
- Example: Programmer makes the code and exposes parameters...
- ...which then get adjusted by the designer

How does your team interact?

- What jobs need to be done before others?
- How do YOU want your team to interact?
- Example: Programmer makes the code and exposes parameters...
- ...which then get adjusted by the designer

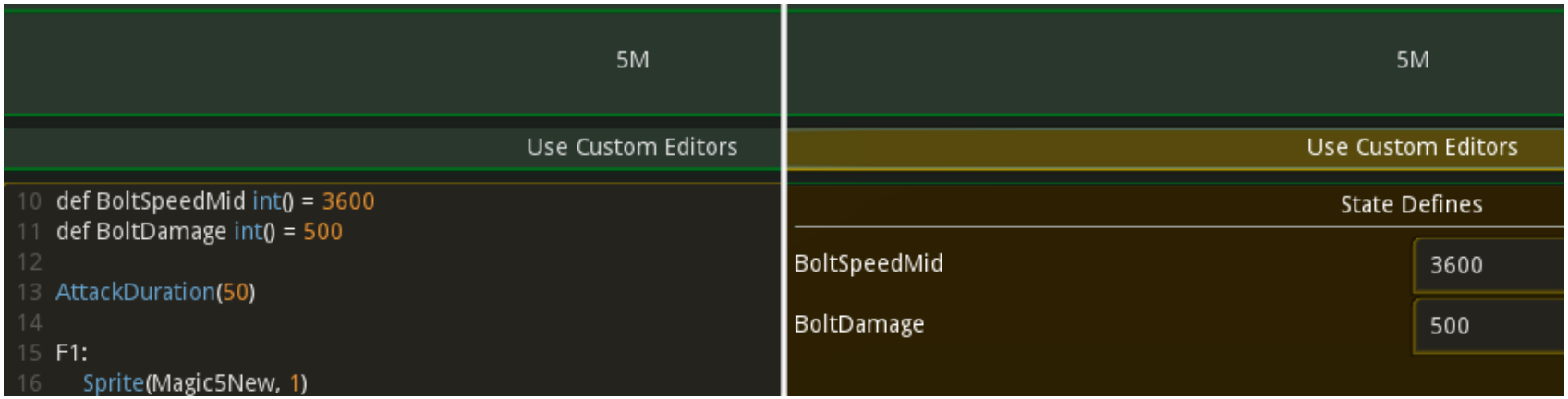
How does your team interact?

- What jobs need to be done before others?
- How do YOU want your team to interact?
- Example: Programmer makes the code and exposes parameters...
- ...which then get adjusted by the designer



How does your team interact?

- What jobs need to be done before others?
- How do YOU want your team to interact?
- Example: Programmer makes the code and exposes parameters...
- ...which then get adjusted by the designer



Opening up the codebase

- One additional possibility: making it public
- Can get tests and contributions
- Not necessarily the same user demographics as your design!

Was it good for Castagne?

Getting there: it allowed for some bug fixes and some features I didn't want to make myself, but also extra work. It was very good for me though.

Opening up the codebase

- One additional possibility: making it public
- Can get tests and contributions
- Not necessarily the same user demographics as your design!

Was it good for Castagne?

Getting there: it allowed for some bug fixes and some features I didn't want to make myself, but also extra work. It was very good for me though.

Opening up the codebase

- One additional possibility: making it public
- Can get tests and contributions
- Not necessarily the same user demographics as your design!

Was it good for Castagne?

Getting there: it allowed for some bug fixes and some features I didn't want to make myself, but also extra work. It was very good for me though.

Opening up the codebase

- One additional possibility: making it public
- Can get tests and contributions
- Not necessarily the same user demographics as your design!

Was it good for Castagne?

Getting there: it allowed for some bug fixes and some features I didn't want to make myself, but also extra work. It was very good for me though.

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Clear Communication

- You need to define your channels.

Clear Roadmap

Tell the why, even if it changes.

Accessible Info

Website, docs, clear signposts.

Gathering Point

Github? Discord? Several?

Get Feedback

Unobstrusive metrics. Castagne survey. Can make it long.

- Try to use tools to make it smoother!

Onboarding

- First impressions are everything! Different people learn differently.

Docs / Text Tutorials

Excellent for programmers and advanced users.

Video Tutorials

Better for artists and procedures like asset import.

In-Engine Tutorials

Stellar for beginners, really useful for showcasing a feature (stays up to date more easily)

Onboarding

- First impressions are everything! Different people learn differently.

Docs / Text Tutorials

Excellent for programmers and advanced users.

Video Tutorials

Better for artists and procedures like asset import.

In-Engine Tutorials

Stellar for beginners, really useful for showcasing a feature (stays up to date more easily)

Onboarding

- First impressions are everything! Different people learn differently.

Docs / Text Tutorials

Excellent for programmers and advanced users.

Video Tutorials

Better for artists and procedures like asset import.

In-Engine Tutorials

Stellar for beginners, really useful for showcasing a feature (stays up to date more easily)

Onboarding

- First impressions are everything! Different people learn differently.

Docs / Text Tutorials

Excellent for programmers and advanced users.

Video Tutorials

Better for artists and procedures like asset import.

In-Engine Tutorials

Stellar for beginners, really useful for showcasing a feature (stays up to date more easily)

Skilling up

Allow your best users to exist

Should provide a path to mastery. Your best users are those that will help you and push your tools and skill further.

- **Beginner**: Basic tutorials and documentation
- **Intermediate**: Can explore the base CASP files
- **Advanced**: Custom tools and modules
- **Expert**: Dive in the code – Architecture is separated at key points

Skilling up

Allow your best users to exist

Should provide a path to mastery. Your best users are those that will help you and push your tools and skill further.

- **Beginner**: Basic tutorials and documentation
- **Intermediate**: Can explore the base CASP files
- **Advanced**: Custom tools and modules
- **Expert**: Dive in the code – Architecture is separated at key points

Skilling up

Allow your best users to exist

Should provide a path to mastery. Your best users are those that will help you and push your tools and skill further.

- **Beginner**: Basic tutorials and documentation
- **Intermediate**: Can explore the base CASP files
- **Advanced**: Custom tools and modules
- **Expert**: Dive in the code – Architecture is separated at key points

Skilling up

Allow your best users to exist

Should provide a path to mastery. Your best users are those that will help you and push your tools and skill further.

- **Beginner**: Basic tutorials and documentation
- **Intermediate**: Can explore the base CASP files
- **Advanced**: Custom tools and modules
- **Expert**: Dive in the code – Architecture is separated at key points

Skilling up

Allow your best users to exist

Should provide a path to mastery. Your best users are those that will help you and push your tools and skill further.

- **Beginner**: Basic tutorials and documentation
- **Intermediate**: Can explore the base CASP files
- **Advanced**: Custom tools and modules
- **Expert**: Dive in the code – Architecture is separated at key points

Conclusion

- Surprisingly practical way to supercharge your workflow and make more game(s)
- Can unlock many opportunities for you and others
- Strong way to shape your gamedev experience and make it more fulfilling!

Conclusion

- Surprisingly practical way to supercharge your workflow and make more game(s)
- Can unlock many opportunities for you and others
- Strong way to shape your gamedev experience and make it more fulfilling!

Conclusion

- Surprisingly practical way to supercharge your workflow and make more game(s)
- Can unlock many opportunities for you and others
- Strong way to shape your gamedev experience and make it more fulfilling!

Conclusion

- Surprisingly practical way to supercharge your workflow and make more game(s)
- Can unlock many opportunities for you and others
- Strong way to shape your gamedev experience and make it more fulfilling!

Conclusion

- Surprisingly practical way to supercharge your workflow and make more game(s)
- Can unlock many opportunities for you and others
- Strong way to shape your gamedev experience and make it more fulfilling!

(Only works after the conference)