
POCKET SIZED
HANDS
Porting Without the Pain
Setting Your Godot Game Up for Success
MADE
WITH
POCKET SIZED HANDS

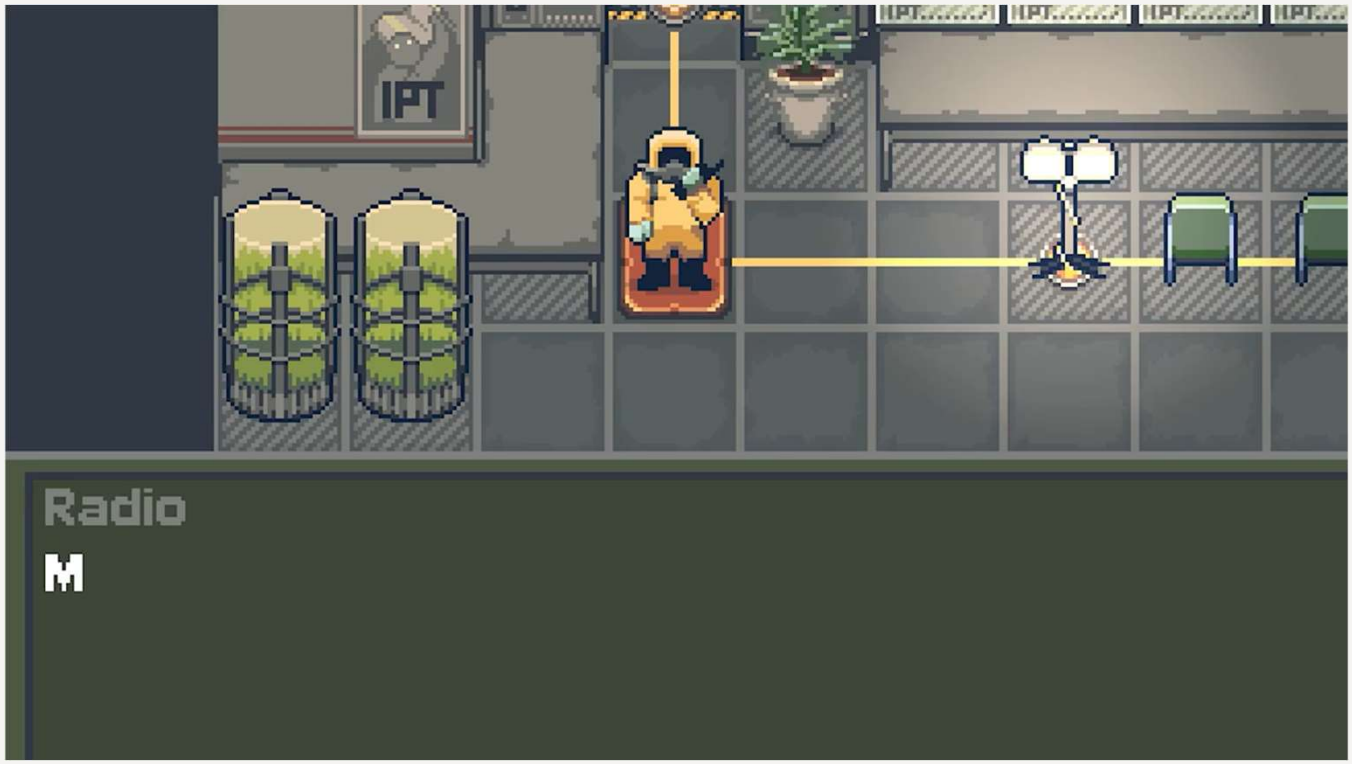
Introduction

- **Me:** Rory Thomson, Director of Engineering
- **Pocket Sized Hands:** Dundee, Scotland, Est. 2017
- **Specialties:** Porting, Co-development, work-for-hire
- **Our Drive:** Solving complex technical challenges
- **2025:** 7 titles shipped, 29 releases, 8 platforms
- **Godot?** Yes please!

Hazard Pay

- **Game:** Hazard Pay
- **Developer:** Smitner Studio
- **Publisher:** Numskull Games
- **Our Role:** Switch and PlayStation 5 ports

numskull GAMES



Pocket Sized Hands

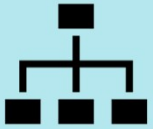
Overview

- **Porting Perspective:** A different way of thinking about game architecture
- **Accelerated Hindsight:** Analyzing completed projects and consequences
- **The ‘Health Check’:** Pocket Sized Hands’ project checklist
- **Today’s Focus:** General best practices applied through the lens of Godot
- **Caveat:** Premature optimization is the root of all evil

Health Check Pt 1

Optimization and performance
What's the lowest target?
Identifying worst case scenarios

Rendering
Forward+ vs Mobile vs Compatability
Fallbacks



Asset Management
Memory restrictions
Load times

Health Check Pt 2

Controller/Input
Compatible/Optimal input mechanisms?

Iconography swapping

Appropriate input management

Localization/Terminology
Localization system in use?

Terminology mapping in place?

Serialization/Save Systems
Expecting instant reads/writes
Saves/Reads too frequently
Many save files

Health Check Pt 3

Audio

Mix loudness

Reliance on audio middleware

UI

Scaling

Navigation

State handling

Miscellaneous

The list could go on

DLC, Online, etc

Pocket Sized Hands

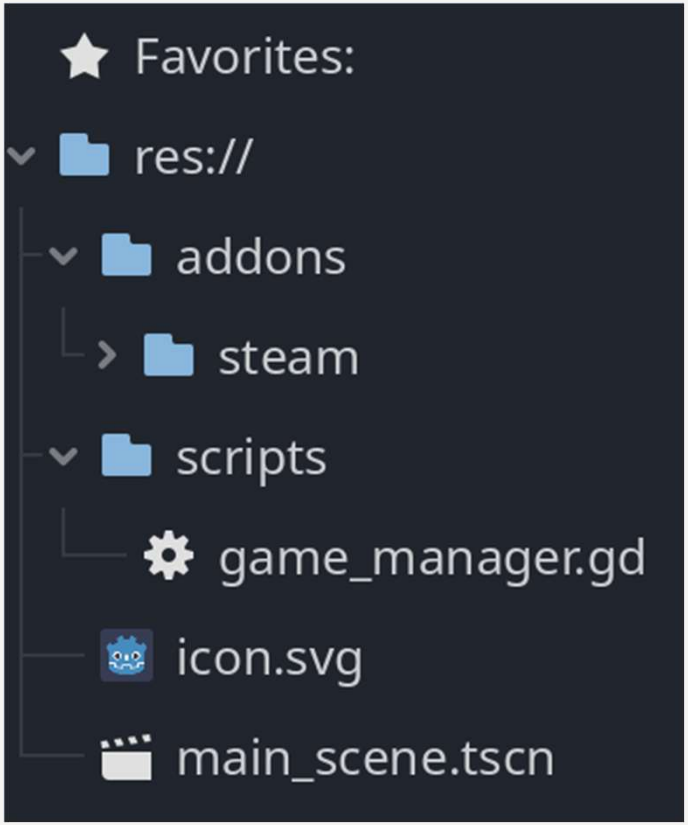
Scripting

- **GDScript** Built for the engine and best supported
- **C#** Relies on .NET runtime, not available on all platforms
- **GDExtension** Need to compile relevant binaries for the target

The bindings below are developed and maintained by the community:

- D
- Go
- Java/Kotlin
- Nim
- Rust
- Swift
- Odin

A Hypothetical – Scripting, Plugins, Achievements




```
1   extends Node
2   class_name GameManager
3
4   func _ready() -> void:
5       Steam.unlock_achievement("my_achievement")
6
```

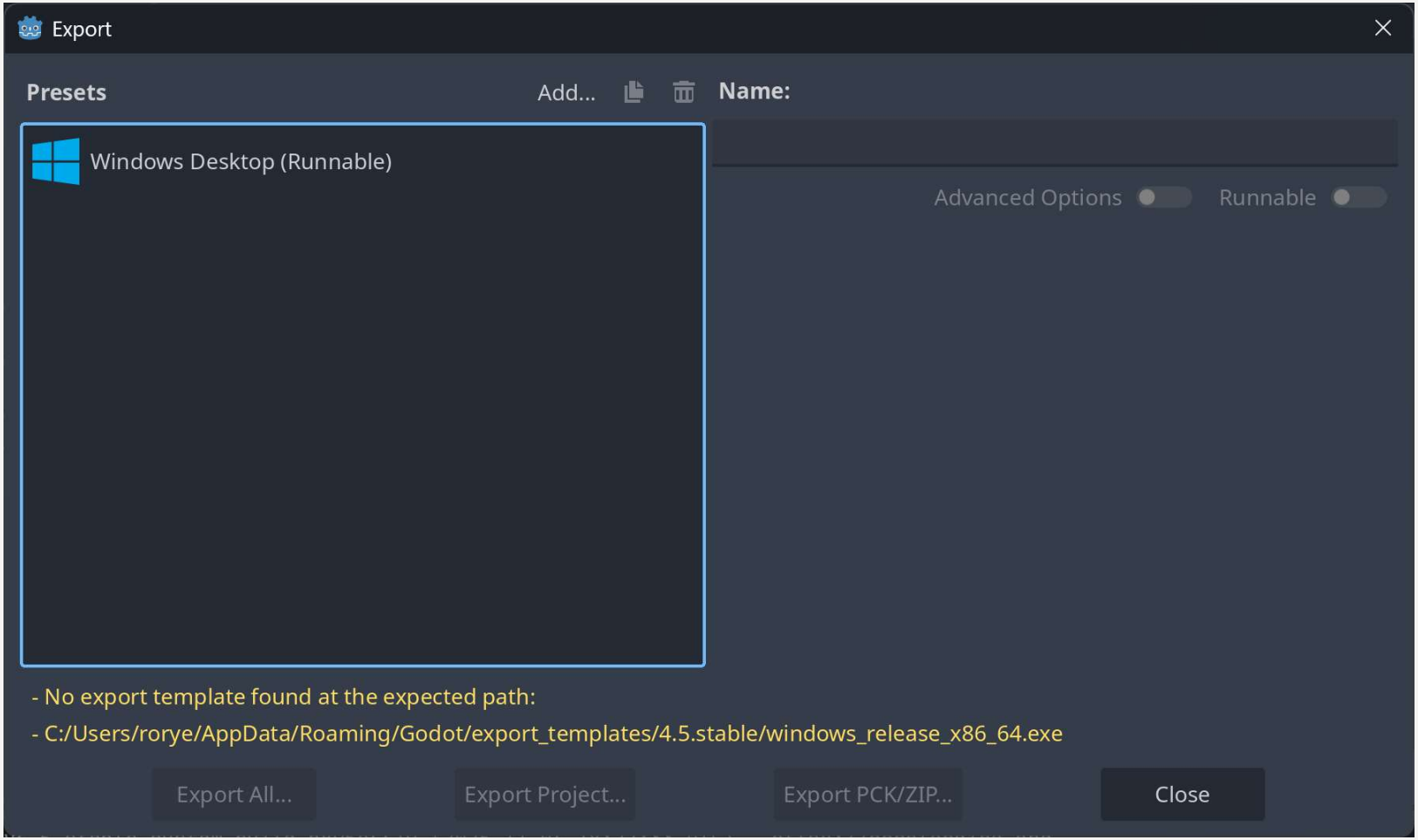
project.godot

D: > GD > my-awesome-project > project.godot

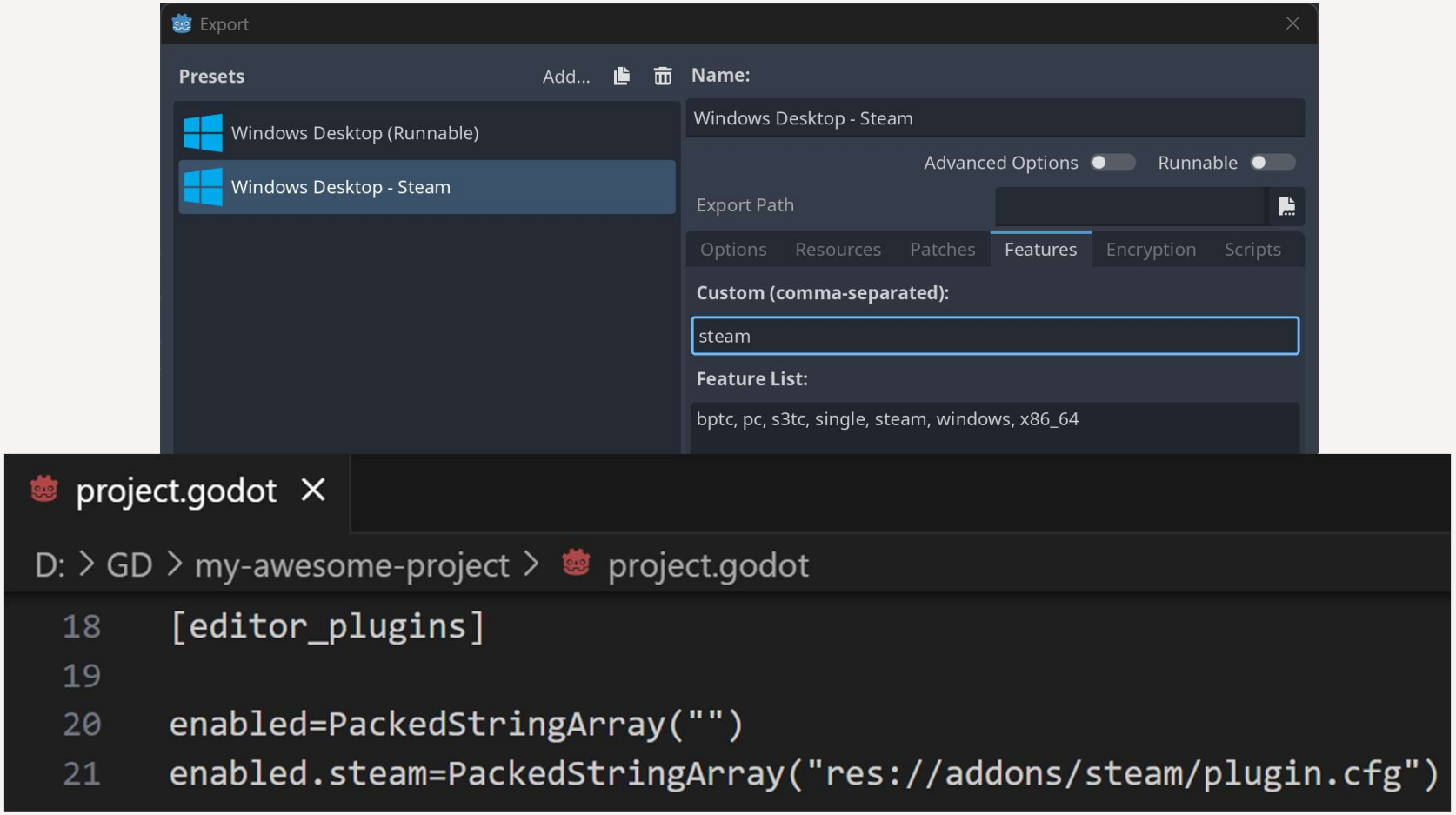
```
18  [editor_plugins]
19
20  enabled=PackedStringArray("res://addons/steam/plugin.cfg")
```

```
Unlocking Steam Achievement: my_achievement
```

A Hypothetical



A Hypothetical – Feature Tags



A Hypothetical

```
SCRIPT ERROR: Parse Error: Could not resolve class "Steam", because of a parser error.
          at: GDScript::reload (res://test.gd:4)
ERROR: Failed to load script "res://test.gd" with error "Parse error".
   at: ResourceFormatLoaderGDScript::load (modules\gdscript\gdscript.cpp:3045)
```

A Hypothetical

```
1  extends Node
2  class_name PlatformBase
3
4  func get_platform_name() -> String:
5      return "Base"
6
7  func unlock_achievement(achievement_name : String) -> void:
8      print("Platform Base Unlock Achievement: " + achievement_name)
9
```

```
1  extends PlatformBase
2  class_name PlatformSteam
3
4  func get_platform_name() -> String:
5      return "Steam"
6
7  func unlock_achievement(achievement_name : String) -> void:
8      Steam.unlock_achievement(achievement_name)
9
```

A Hypothetical

```gdscript
extends Node
class_name PlatformManager

enum PlatformType { NULL, STEAM }

static var platform_instance : PlatformBase

static func get_platform_type() -> PlatformType:
    if OS.has_feature("steam"):
        return PlatformType.STEAM

    return PlatformType.NULL

func _enter_tree() -> void:
    var platform_script
    match get_platform_type():
        PlatformType.STEAM:
            platform_script = load("res://scripts/platform/steam/platform_steam.gd")
        _:
            platform_script = load("res://scripts/platform/platform_base.gd")

    platform_instance = platform_script.new()
    add_child(platform_instance)
    print("Initialized Platform: " + platform_instance.get_platform_name())

static func unlock_achievement(achievement_name : String) -> void:
    if not platform_instance:
        pass

    platform_instance.unlock_achievement(achievement_name)

```

A Hypothetical

```
1   extends Node
2   class_name GameManager
3
4   func _ready() -> void:
5       PlatformManager.unlock_achievement("my_achievement")
6
```

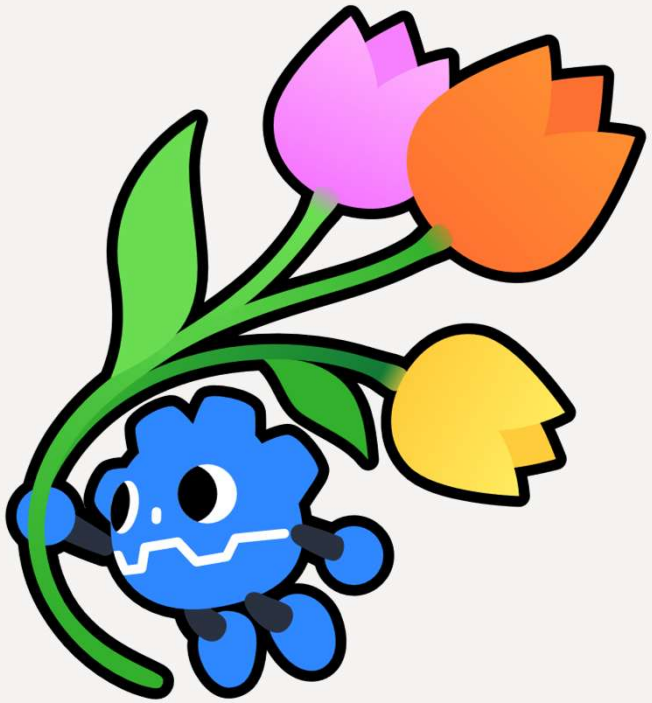
A Hypothetical

```
Initialized Platform: Steam
Unlocking Steam Achievement: my_achievement
```

```
Initialized Platform: Base
Platform Base Unlock Achievement: my_achievement
```

A Hypothetical – Next Steps

- **Another target/platform?** Add a new feature tag, platform implementation and you're on your way.
  - Epic Games Store, Google Play, Console platform
- **Other Applications** Terminology swapping, UI Specialization



Pocket Sized Hands

Rory Thomson

rory@pocketsizedhands.com

Thank you

POCKET SIZED HANDS