# Design and implementation of the Meta Casanova 3 Compiler front-end

Jarno Holstein

July 1, 2016

# Introduction

## In this presentation

- Introduction
- Research question
- Sub questions
- Results
- Conclusion
- Demo

# Introduction

## Motive

- Video game development
- Casanova
- Meta Casanova
- Front-end of the compiler

Meta Casanova became usefull for more then game development

# Research question

## Main question

How to develop a maintainable and expandable front-end and type checker for the Meta Casanova programing language?

# Research question

## Requirements

- Correct
- Maintainable and expandable
- Descriptive error messages

# Sub questions

- Syntactic properties question
- Parser question
- Type system question

# Syntactic properties question

What properties does MC have that the front-end needs to process?

# Syntactic properties question

## Keywords

MC

- Func
- Data
- TypeFunc
- TypeAlias
- ->
- =>
- #>
- import
- inherit
- Module
- (\ )

C#

abstract as base bool break byte case catch char checked class const continue decimal default delegate do double else enum event explicit extern false finally fixed float for foreach goto if implicit in int interface internal is lock long namespace new null object operator out override params private protected public readonly ref return sbyte sealed short sizeof stackalloc static string struct switch this throw true try typeof uint ulong unchecked unsafe ushort using virtual void volatile while

# Syntactic properties question

## Divided features

- Data and Func
- .NET
- TypeFunc
- Module

# Parser question

How to develop a maintainable and expandable parser for MC?

# Parser question

## What is a parser

- Takes text
- Data structure
- Syntactic errors

# Parser question

## How to make a parser

- Parser generators
- Parser monad

# Parser question

## Parser generator

- Program
- Syntax description -> parser

# Parser question

## Parser generator

Pros

- Fast setup
- Fast parsing

Cons

- Error messages

# Parser question

## What is a parser monad?

- Takes a list
- Outputs a data structure
- Returns an error when fails

# Parser question

## Parser combinators

- Takes parser monads
- Returns a parser monad
- Try multiple parser monads
- Catches parser errors

- character_parser until end_of_line_parser
- digit_parser and dot_parser and digit_parser

# Parser question

## Error handling within the parser monad

Concat all the errors

1. Slow
2. Get all the errors and thus none

Give only the last viable error

1. Fast
2. possibility to lose error information
3. possibility to get incorrect error information

Priority for errors

1. Fast
2. More accurate error information
3. possibility to get incorrect error information

# Parser question

## Parser monad

Pros

- Custom error system
- Parser combinators
- Easy to maintain and expand

Cons

- Complex setup
- Slow parsing

# Parser question

## Strategy

| **Feature** | **Parser modules** | **Type checker** |
|---|---|---|
| Data and func | Definitions & declarations of Func and Data | Minimal type checker |
| Dotnet | | DotNet types |
| TypeFunc | Definitions & declarations of TypeFunc and TypeAlias | Compile time interpretation |
| Module | | Complex inheritance system |

# Type system question

How to apply type systems to MC?

# Type system question

## Type system of MC

Needs to:

- Compare types and sub types
- Inference types
- Create types

# Type system question

## Type checker

- Normalized input and output
  - One representation of program
  - Back-end interface
- Modular
- Runtime and compile time

# Change of plans

- Request for dependent types
- Complexer type checker
- Research member
- Incompatible
- New parser

# Results

## Requirements

- Correct
- Maintainable and expandable
- Descriptive error messages

# Conclusion

- Requirements are met
- Working front-end
- Helped the research team
- Programming industry

# Demo

# Questions

?