GRIMM

Choose-Your-Own-Story Language

Mike Lenner
Billy Liu
Mariya Nomanbhoy
Becky Plummer

What is GRIMM

- Named for famous storytellers – Grimm Brothers
- Designed to make creating an interactive, first-person role playing game easy and intuitive
- No programming experience required – language based on free-form English
- Reallocates the majority of effort where it belongs – the story writing

Language Outline

- First section of GRIMM programs are declarations
- Developer declares scenes within a story. Each scene contains standard set of properties
  - Name ("The Living Room")
  - Description ("You are standing in a room with green walls")
  - Picture ("room.jpg")
  - Items (couch, notebook)
  - Exits (hallway, kitchen)
- Developer may also declare items and characters

GRIMM Example - Declarations

```
scene castleGate
scene courtYard

castleGate name “The Castle Gate”
castleGate description “You are standing in front of a large
   wooden gate.  The gate is locked and there are two soldiers
   preventing your passage.”
castleGate exit courtYard

courtYard name “Castle Court Yard”
courtYard description “You have entered the castle court yard.
   You see that a joust tournament is about to begin.”
courtYard contains item lance
courtYard contains item helmet

character Merlin
Merlin holds item staff
```

Language Outline

- Second part of GRIMM program is the actual story using these declared scenes, characters and items.
- Control flow constructs guide user through declared scenes based on user input.
- Conditionals test the state of the user (what scene is occupied, what items are owned).

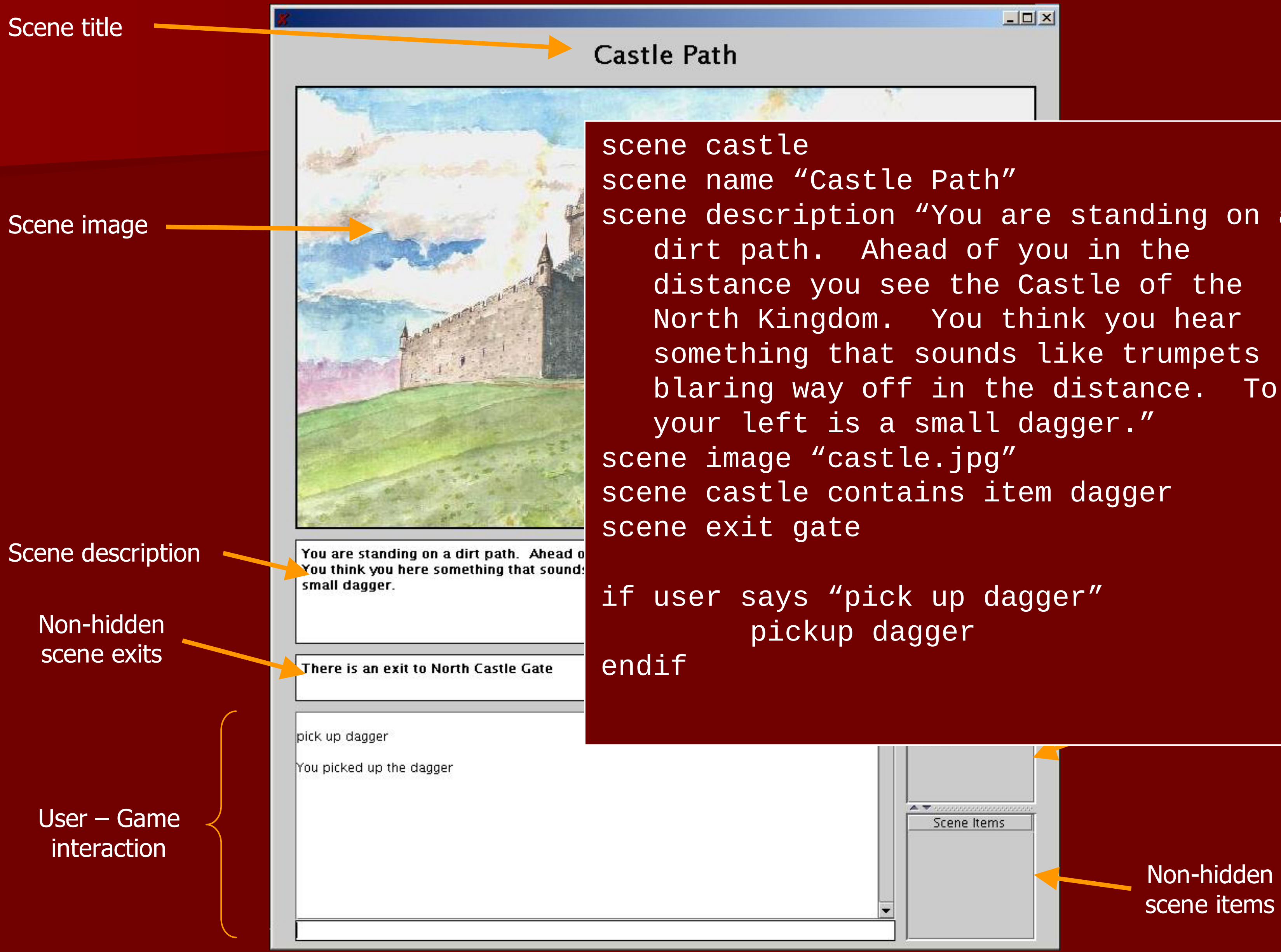
GRIMM Example – Control Flow

```
while user inside dungeon
   read user input
   if user says “unlock cage door” and not user has key then
        say “You cannot unlock the door without the key.”
   otherwiseif user says “unlock cage door” and user has key then
        say “The door is unlocked and you slip through the cage
             unnoticed.  You move quickly to the treasure room
             next door.”
        goto treasureRoom
   otherwiseif user says “open treasure chest” then
        say “You open the chest and you see a silver key.”
        read user input
        if user says “take key” then
               user pickup key
               say “You took the key”
        otherwise
               say “The chest has closed shut on its own.”
        endif
   endif
endwhile
```

Game Play

- Story is played out by displaying current scene to the user, then waiting for input.
- Many such games are text based – GRIMM creates graphical based applications.
- User interacts with a GUI console
  - – shows the current state of the game as well as allows for user interaction.

GUI Components



Castle Path

You are standing on a dirt path. Ahead o
You think you here something that sound
small dagger.

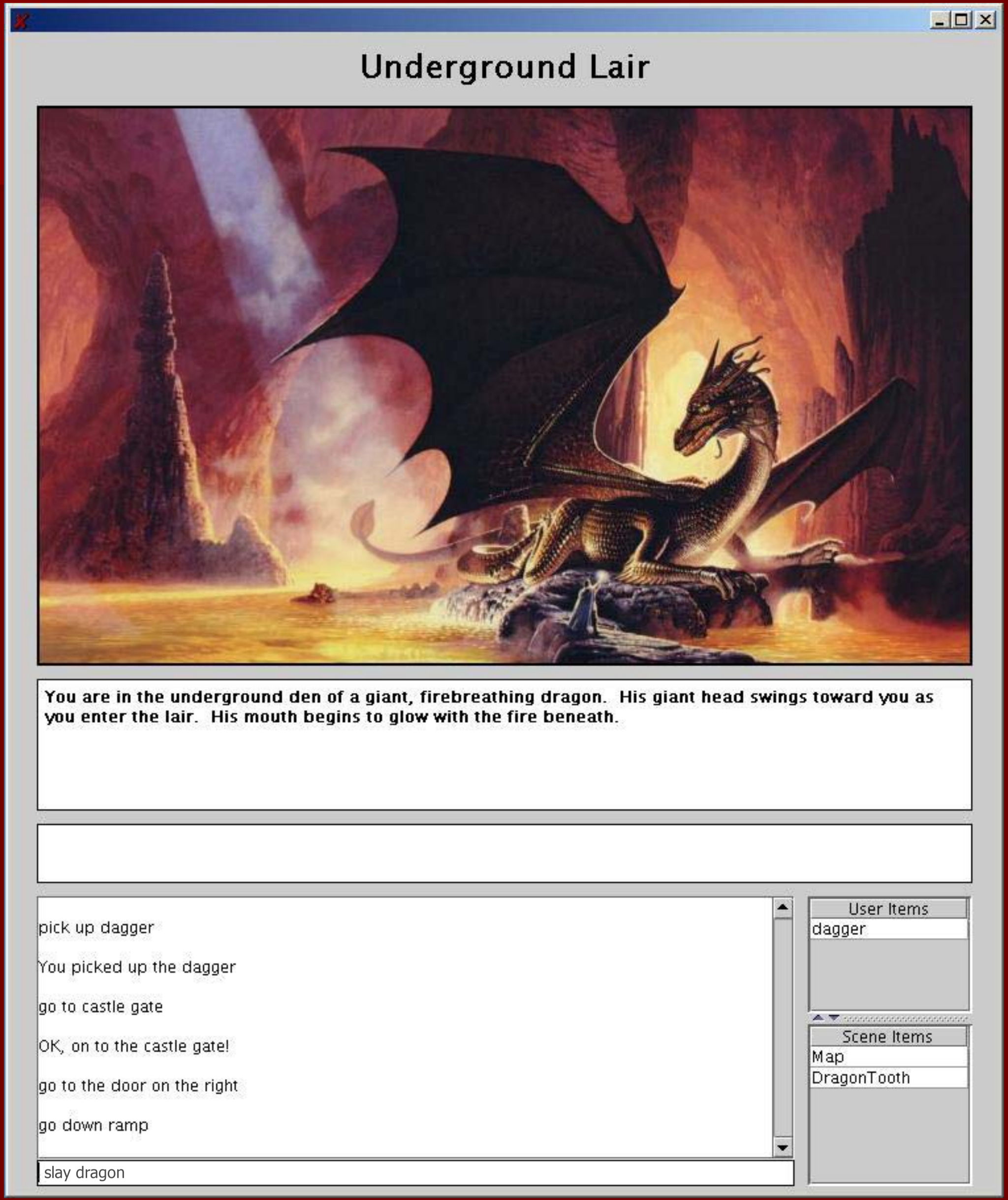
There is an exit to North Castle Gate

pick up dagger

You picked up the dagger

```
scene castle
scene name “Castle Path”
scene description “You are standing on a
   dirt path.  Ahead of you in the
   distance you see the Castle of the
   North Kingdom.  You think you hear
   something that sounds like trumpets
   blaring way off in the distance.  To
   your left is a small dagger.”
scene image “castle.jpg”
scene castle contains item dagger
scene exit gate

if user says “pick up dagger”
        pickup dagger
endif
```

Scene Update



Underground Lair

You are in the underground den of a giant, firebreathing dragon. His giant head swings toward you as you enter the lair. His mouth begins to glow with the fire beneath.

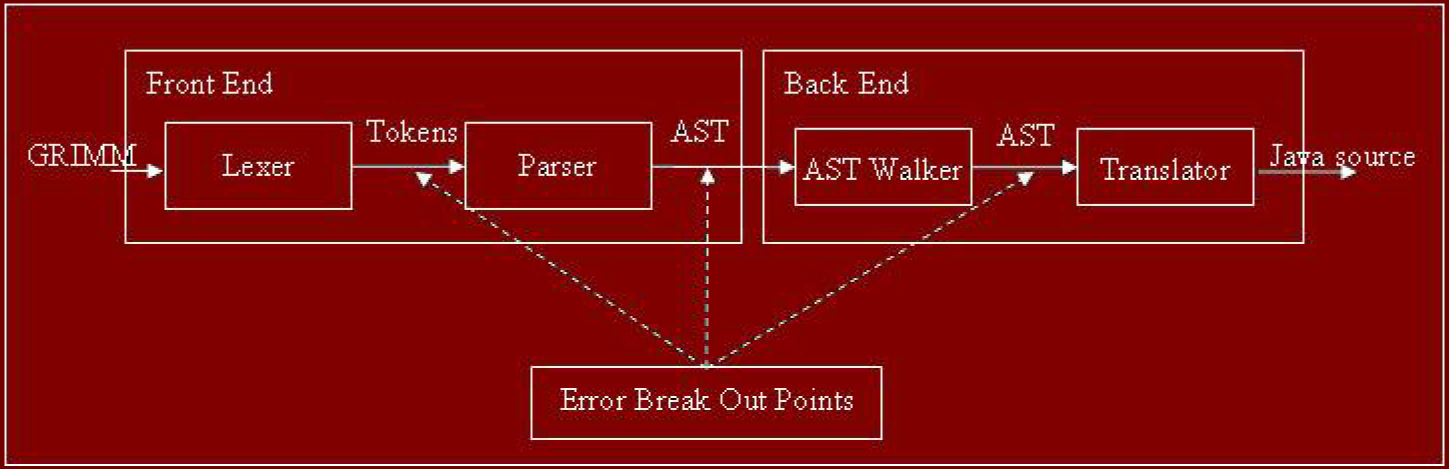
```
pick up dagger

You picked up the dagger

go to castle gate

OK, on to the castle gate!

go to the door on the right

go down ramp
```

```
slay dragon
```

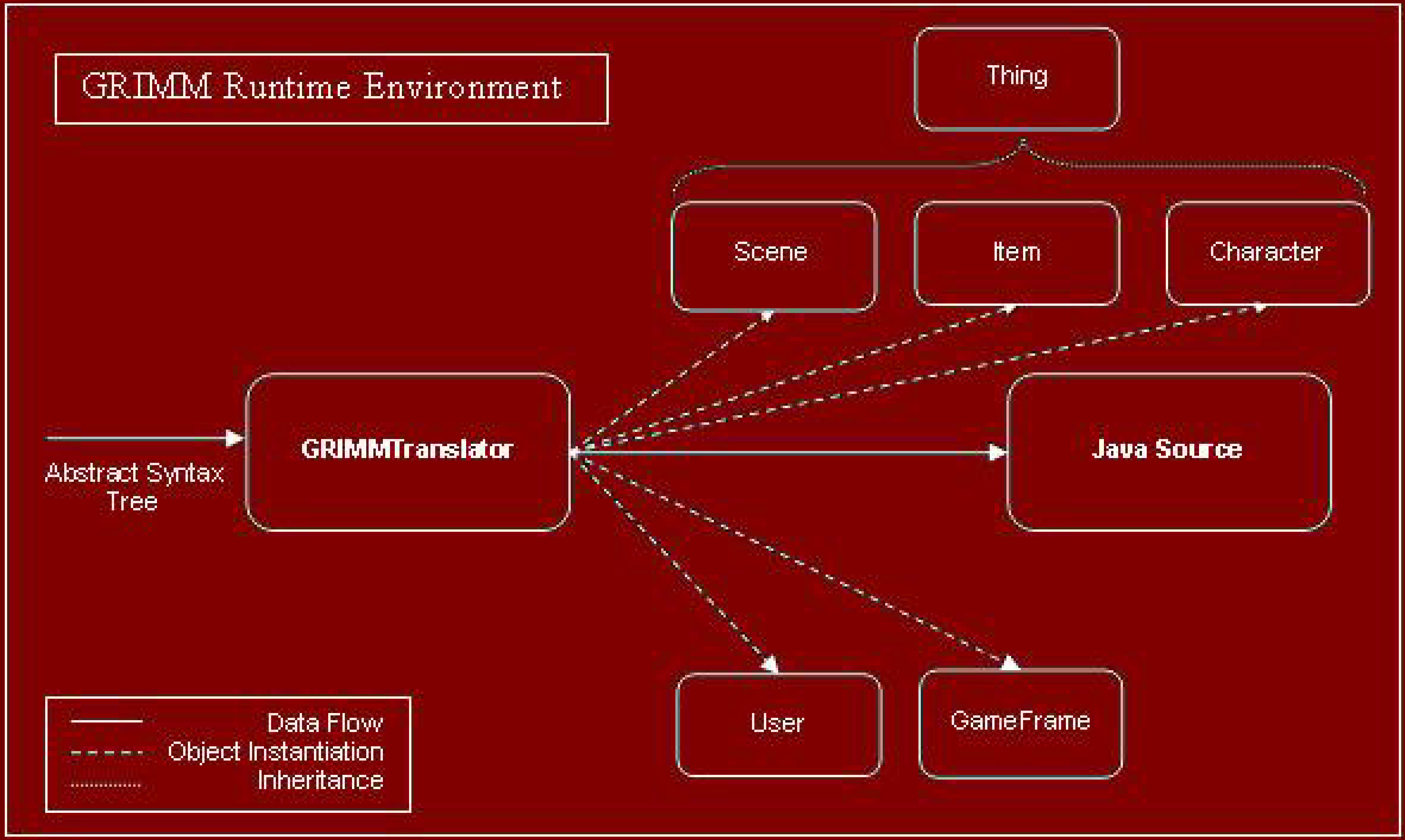
GRIMM Implementation

- Lexer – separate text into token stream
- Parser – builds syntactically correct AST
- Walker – builds symbol table while checking for semantic errors
- Translator – walks AST generating Java code based on each node
- Runtime Environment – Java classes used by Translator in creating application code

GRIMM Implementation



Runtime Environment



Lessons Learned

- Understand big picture before coding begins
- Split up responsibilities earlier in the project
- Know the scope of your language from the beginning and try not to deviate
- Difficult to maintain free-from English syntax when building the compiler

Summary

- GRIMM is a useful tool for the storywriter
  - Less time spent on coding, more on developing storylines.
  - A lot can be accomplished without a lot of code.
- Maintained somewhat intuitive syntax
- Extremely successful team collaboration