Software Engineering Spring 2010

Duel Reality

Josh Kilgore
Obi Atueyi
Tom Calloway
Ye Tian

Summary

- Overview
- Modules
  - Graphics (Tom Calloway)
  - User Interface (Obi Atueyi)
  - Game Mechanics and AI (Josh Kilgore)
  - Database (Ye Tian)
- Conclusion
- Questions

The Overview

TEAM GOLD

- Game Description
  - Details
  - Game flow, battle flow
- Tools
- Module Breakdown

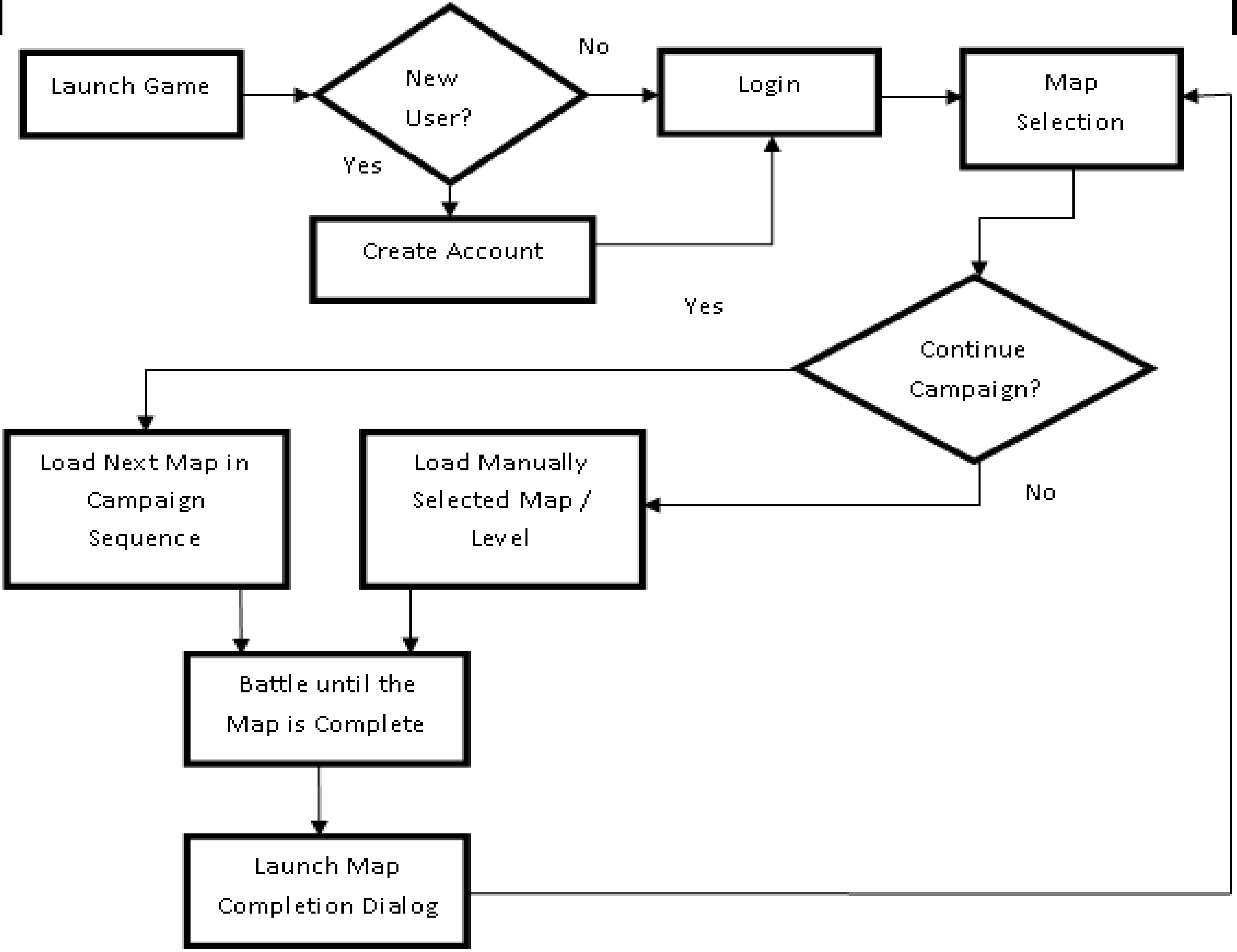
Duel Reality: the Game

- Total Game play experience
  - 2D
  - Turn based
  - Strategy battle simulation
  - 1 Player w/ Ai opponent
  - (maybe 2)
  - Upgradable units
  - 7 - level Campaign and Free Battle Modes
  - Save / Load functionality
  - Amazing Graphics and Sound
  - Dynamic Action Points Game play

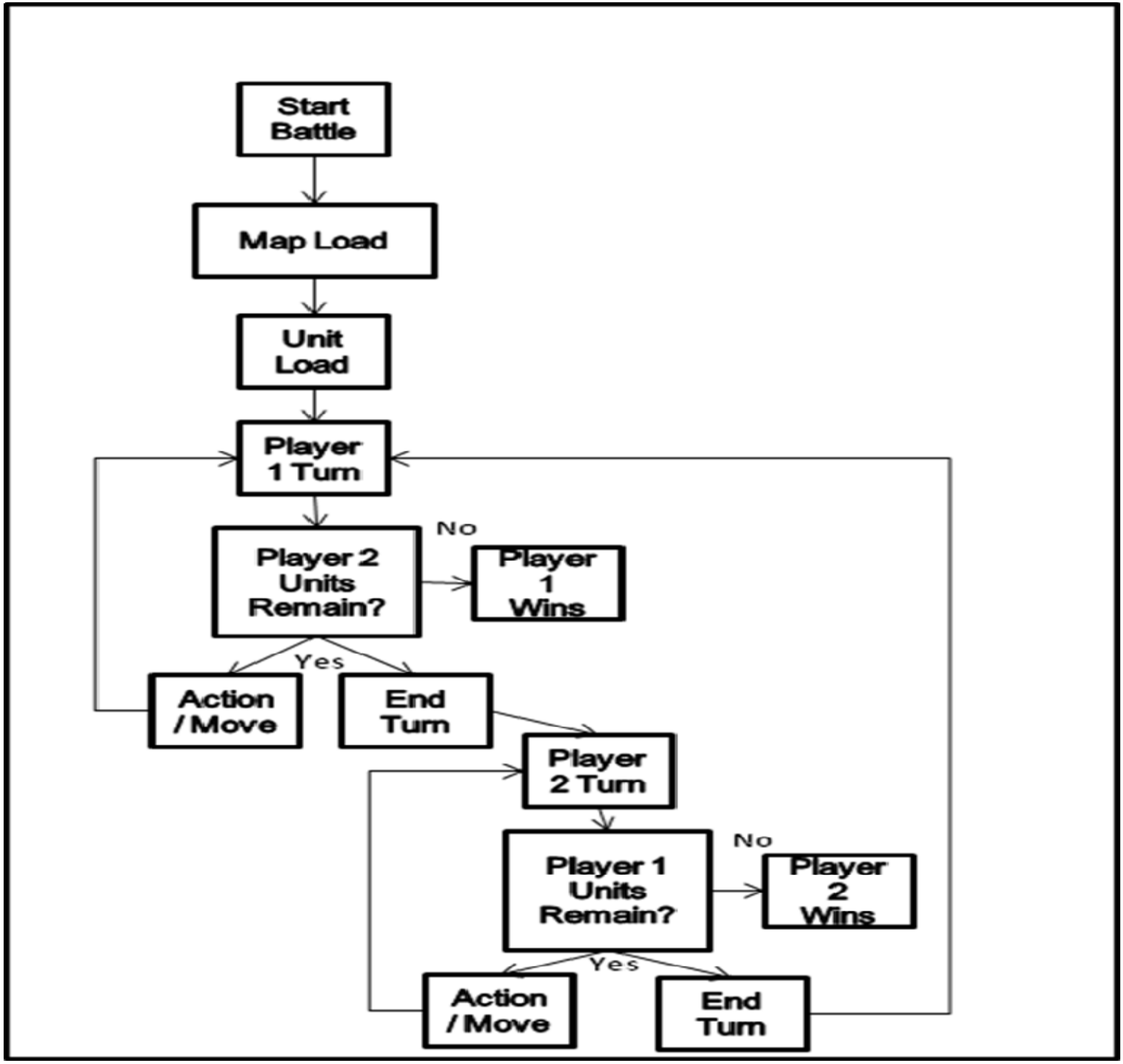
X			O		
	X				
X				O	
					O

Main Game

TEAM GOLD


Launch Game
New User?
No
Yes
Login
Map Selection
Create Account
Yes
Continue Campaign?
No
Load Next Map in Campaign Sequence
Load Manually Selected Map / Level
Battle until the Map is Complete
Launch Map Completion Dialog

Battles

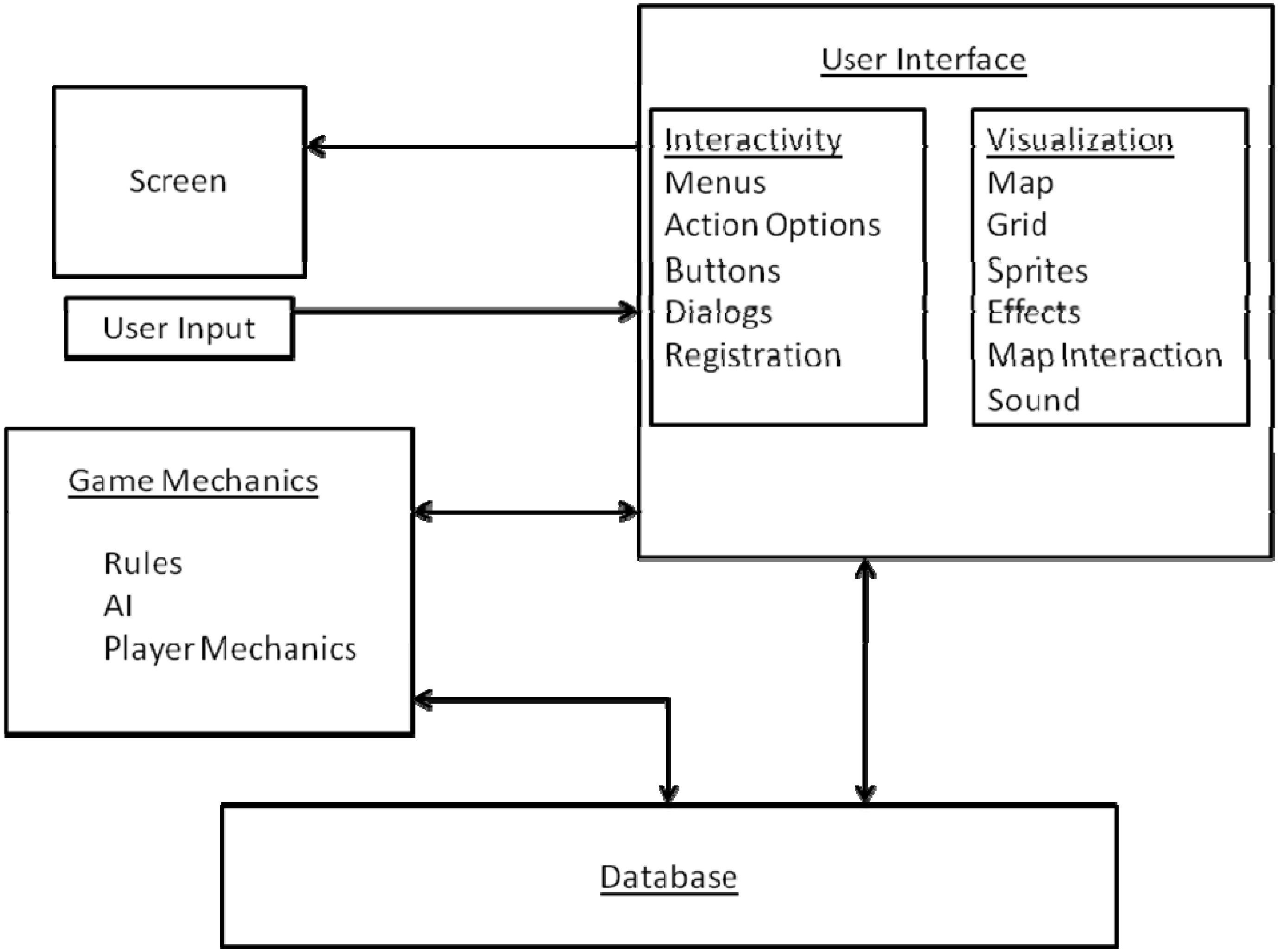


Basically:

Tools

- Qt Development Platform
- OpenGL Graphics
- SQLite Database
- C++

Modules



Screen

User Input

User Interface

Interactivity
Menus
Action Options
Buttons
Dialogs
Registration

Visualization
Map
Grid
Sprites
Effects
Map Interaction
Sound

Game Mechanics

Rules
AI
Player Mechanics

Database

Overview Summary

- Game Description
  - Details
  - Game flow, battle flow
- Tools
- Module Breakdown
  - Graphics
  - UI
  - Game Mechanics
  - Database

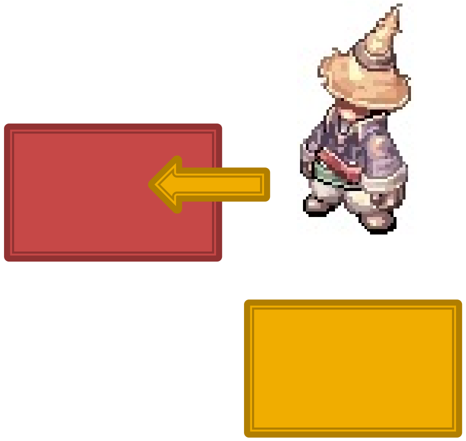
X			O		
	X				
X				O	
					O

Game Graphics Module

Presenter: Tom Calloway

TEAM GOLD

- What it is.
  - Purpose and overview.
- What it does.
  - Key functionality.
- How it is implemented.
  - Tools & Architecture.
  - Challenges.
  - Unit Test & Verification.

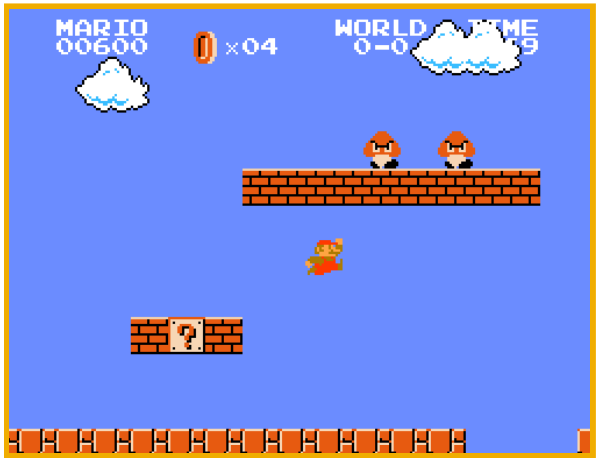


Game Graphics Module

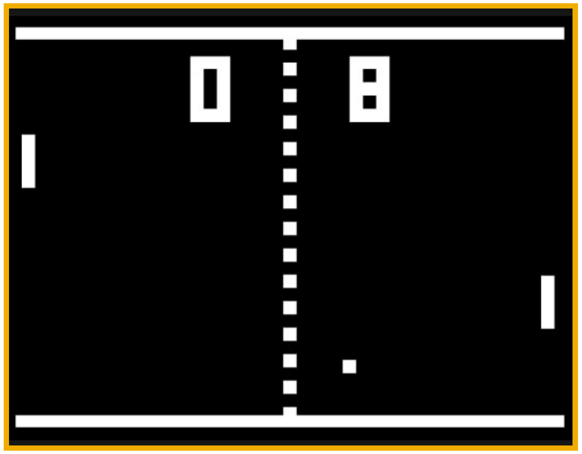
Purpose & Overview (What it is)

TEAM GOLD

- The 2D Graphics & Sound Effects!



Super Mario Bros. - 1985

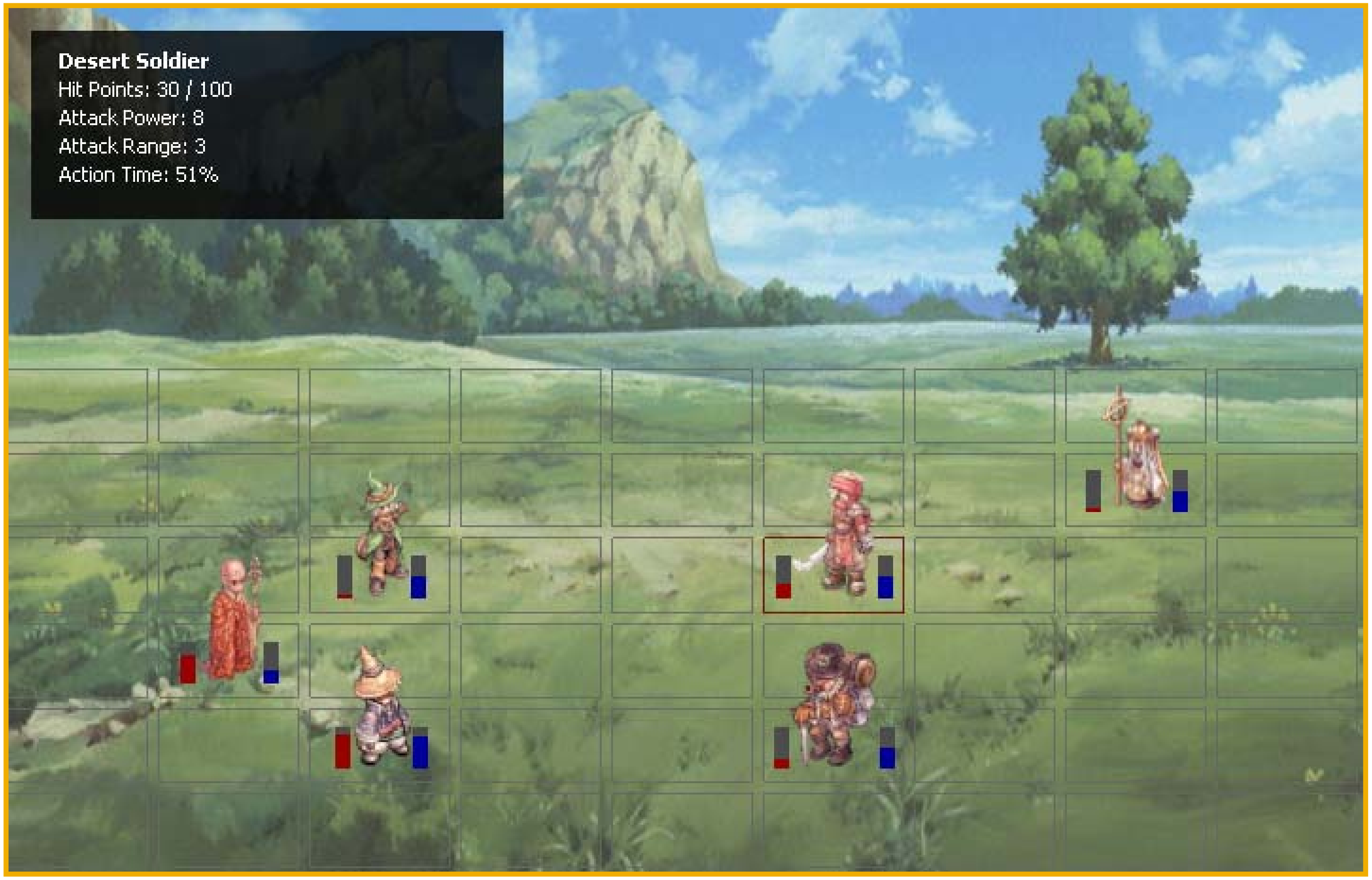


Pong - 1972

Game Graphics Module

Functionality (What it does)

TEAM GOLD



Desert Soldier
Hit Points: 30 / 100
Attack Power: 8
Attack Range: 3
Action Time: 51%

Game Graphics Module

Implementation (Tools)

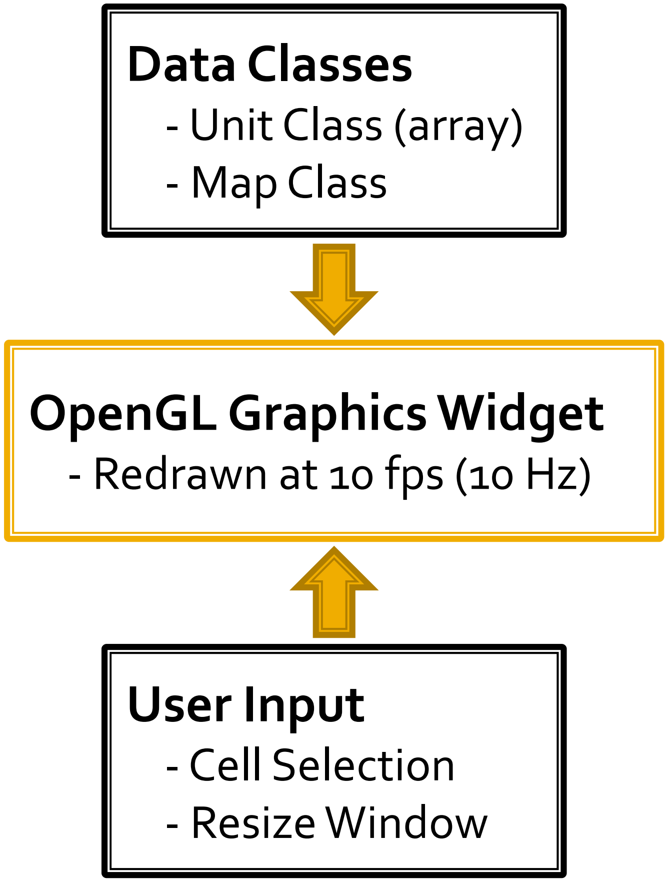
- Some Possibilities
  - Direct image manipulation
  - Microsoft DirectX Technology
  - OpenGL (Open Graphics Libraries)

- OpenGL Selected
  - Cross-platform
  - Simple Qt Integration
  - Free ($$)

Game Graphics Module

Implementation (Architecture)

- How it works.
  - Receives data from other modules.
  - Content is loaded from hard drive.
  - Allows events triggered by external code modules and users (e.g., move, select, attack, remove, add) .



Data Classes
- Unit Class (array)
- Map Class

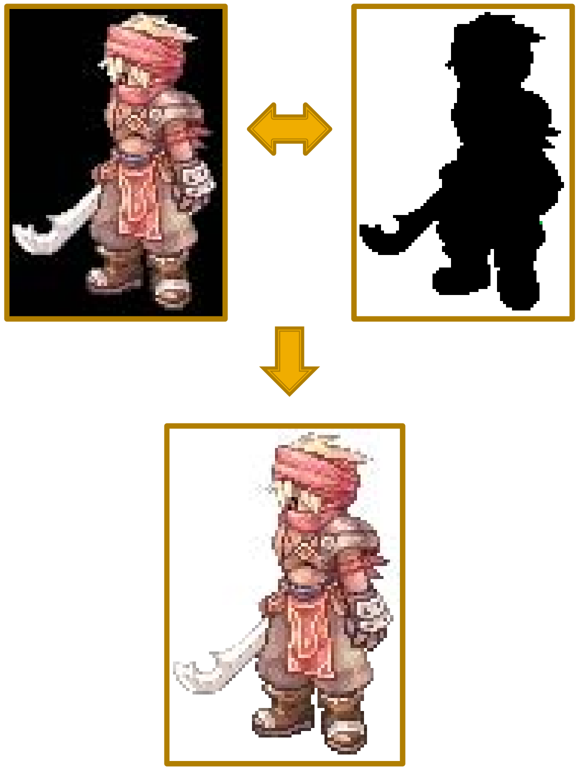
OpenGL Graphics Widget
- Redrawn at 10 fps (10 Hz)

User Input
- Cell Selection
- Resize Window

Game Graphics Module

Implementation (Challenges)

- Transparency
  - Bit masking of multiple images.
- Mouse Interactions
  - Coordinate calculations.
- OpenGL Familiarity
  - Know what is available.
  - Know how functions work.
  - Understand quirks.
- Many Others…

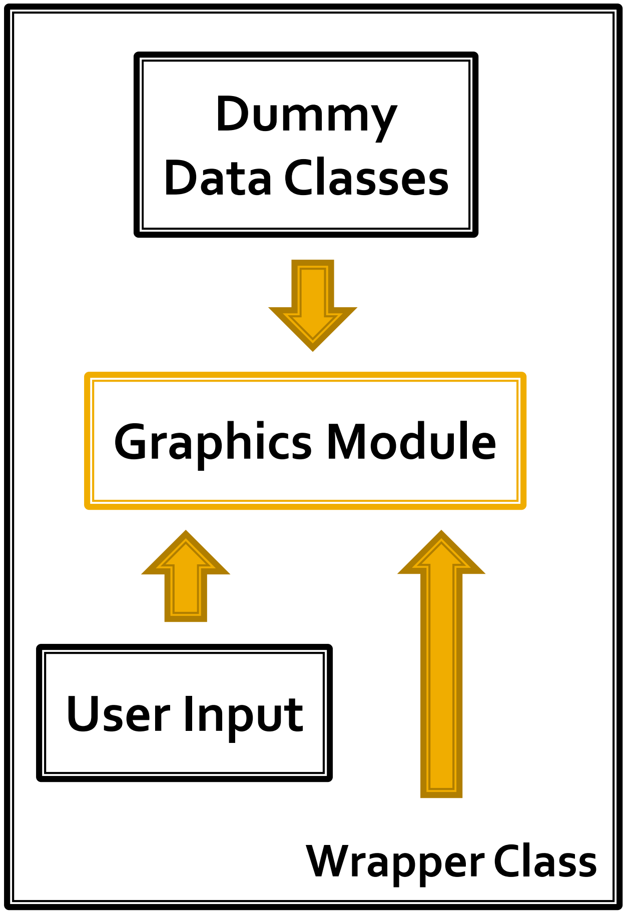


Game Graphics Module

Unit Test & Verification

TEAM GOLD

- Create "Dummy" Data Classes
  - Tweak data and observe the results.
- Create "Wrapper" Class
  - Calls and exercises functions.



Game Graphics Module

Summary

- Graphics vs. Mechanics
  - 2D Mechanics = 2D Overhead Graphics.

- Tools / Architecture Choices Important
  - Avoid future frustration.
  - Create a solid game.

User Interface Module

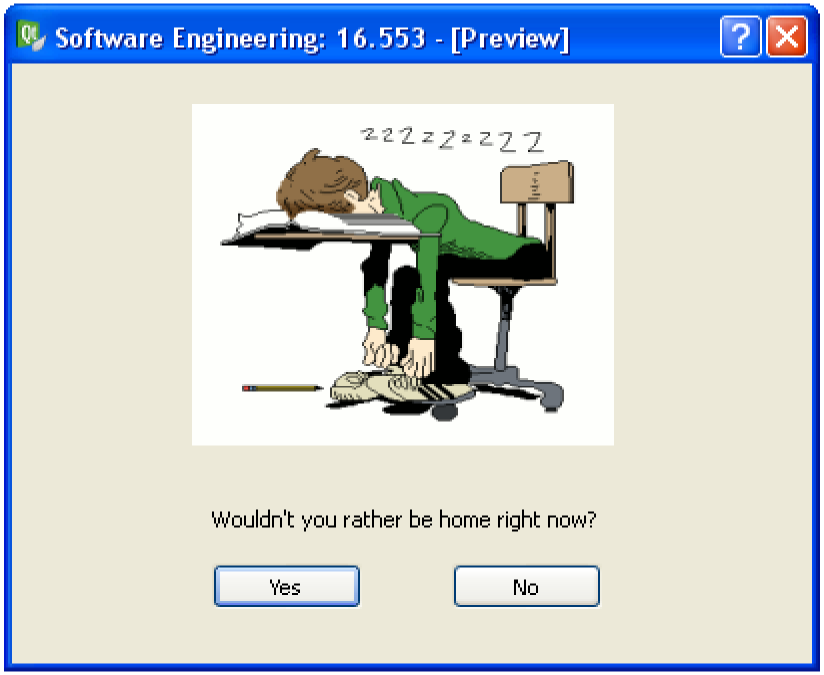
Presenter: Obi Atueyi

- Function
- Architecture
  - Module Interactions
  - Mainwindow
  - Application Interaction Items
- Implementation
  - Tools
  - Class Diagrams
  - Qt Classes
    - QWizard & QWizardPage Classes
  - Unit Test & Verification
- Challenges
- Summary

User Interface Module

Function

- Provide the framework for user interaction with the application



Software Engineering: 16.553 - [Preview]

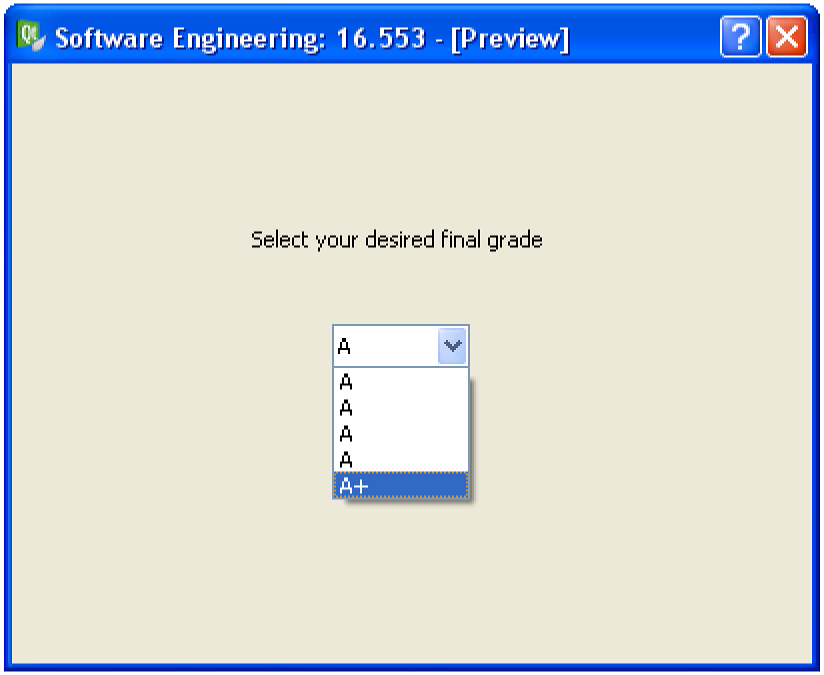
Wouldn't you rather be home right now?

User Interface Module

Function

TEAM GOLD

- Provide the ability for user to choose desired settings



Select your desired final grade

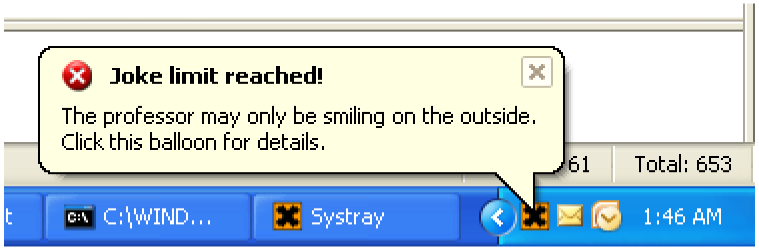
A

A
A
A
A
A+

User Interface Module

Function

- Provide status or error messages



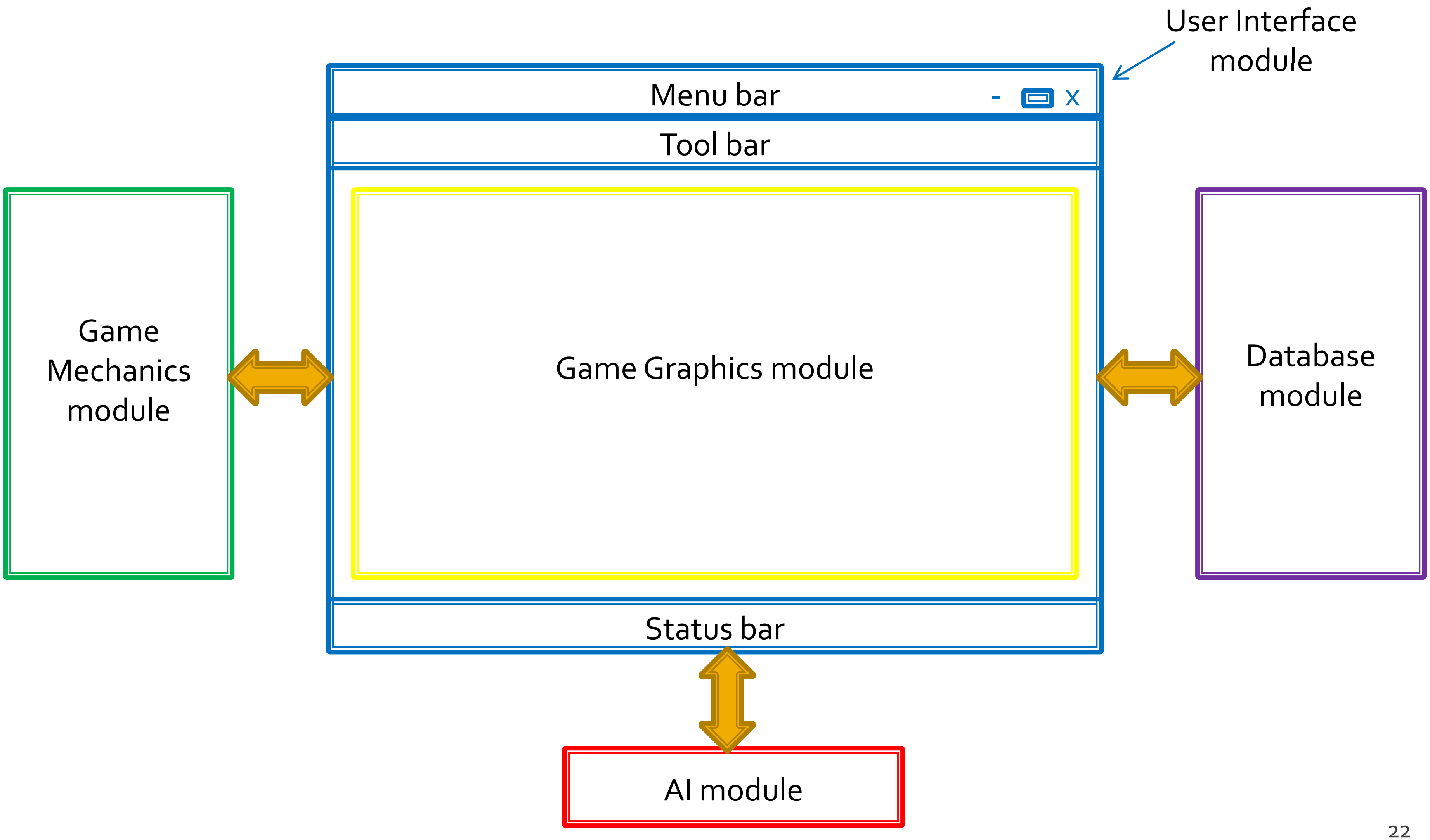
Joke limit reached!

The professor may only be smiling on the outside.
Click this balloon for details.

User Interface Module

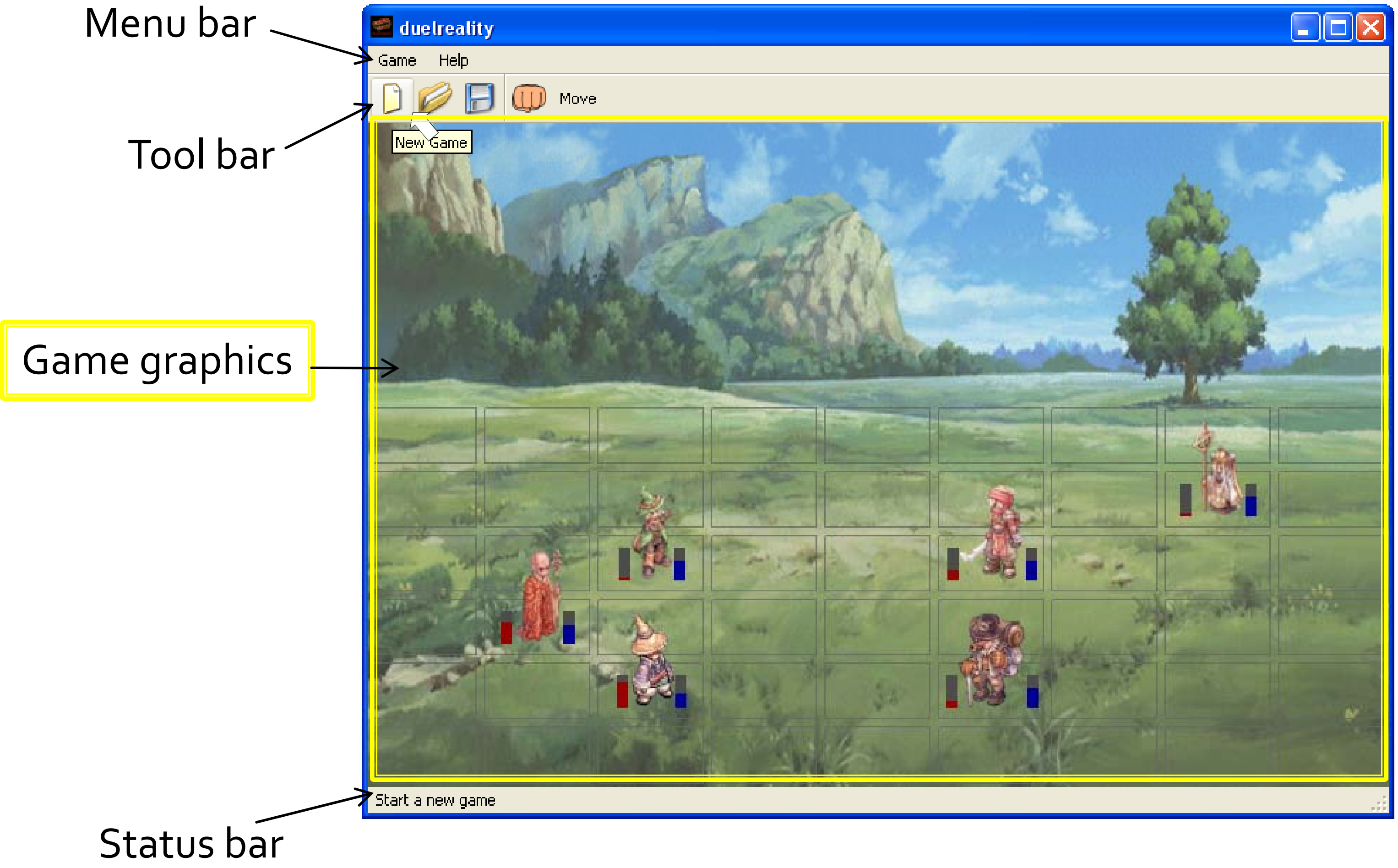
Architecture: Module Interactions

TEAM GOLD



User Interface Module

Architecture: Mainwindow



Start a new game

Menu bar

Tool bar

Game graphics

Status bar

User Interface Module

Architecture: Application Interaction Items

- Menu Items
  - New Game
  - Load Game
  - Save Game
  - Restart Game
  - Quit Game
- Toolbar Items
  - Attack
  - Move
  - End Turn
- Status Bar Items
  - Tool tips

User Interface Module

Implementation: Tools

Considered Tools

- Qt
- wxWidgets
- MFC

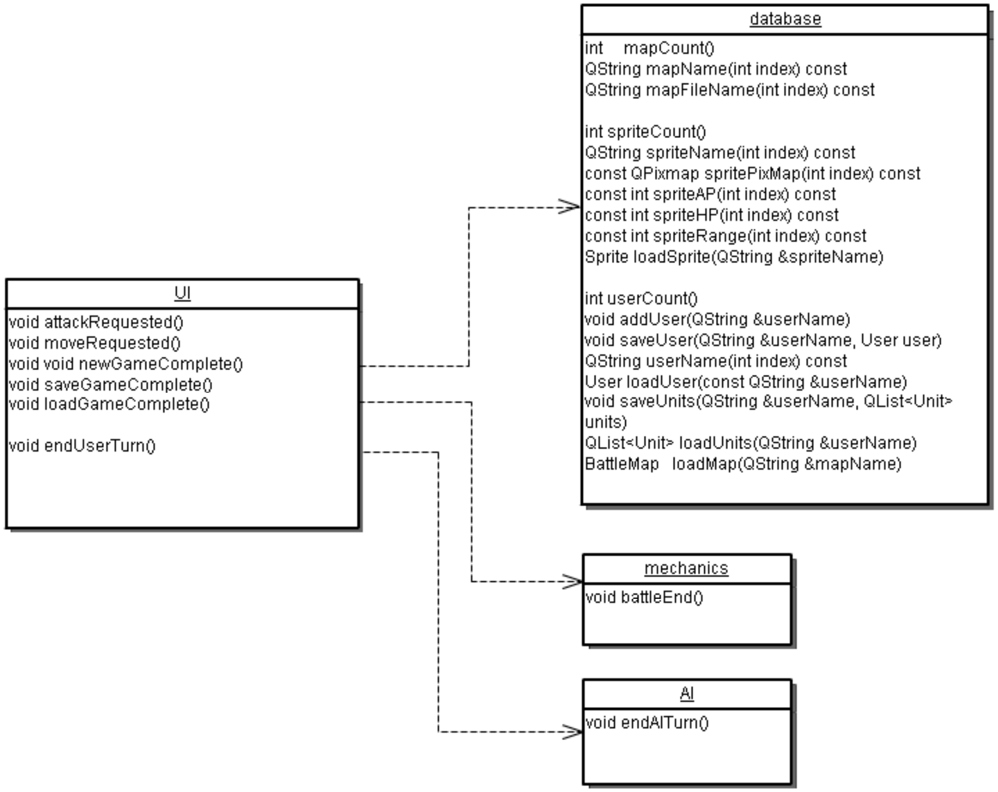
Qt Selected

- Cross-platform
- Non-GUI features (SQL database)
- Meta-object compiler (object macros)

User Interface Module

Implementation: Class Diagram

TEAM GOLD



UI

void attackRequested()
void moveRequested()
void void newGameComplete()
void saveGameComplete()
void loadGameComplete()

void endUserTurn()

database

int mapCount()
QString mapName(int index) const
QString mapFileName(int index) const

int spriteCount()
QString spriteName(int index) const
const QPixmap spritePixMap(int index) const
const int spriteAP(int index) const
const int spriteHP(int index) const
const int spriteRange(int index) const
Sprite loadSprite(QString &spriteName)

int userCount()
void addUser(QString &userName)
void saveUser(QString &userName, User user)
QString userName(int index) const
User loadUser(const QString &userName)
void saveUnits(QString &userName, QList<Unit> units)
QList<Unit> loadUnits(QString &userName)
BattleMap loadMap(QString &mapName)

mechanics

void battleEnd()

AI

void endAITurn()

User Interface Module

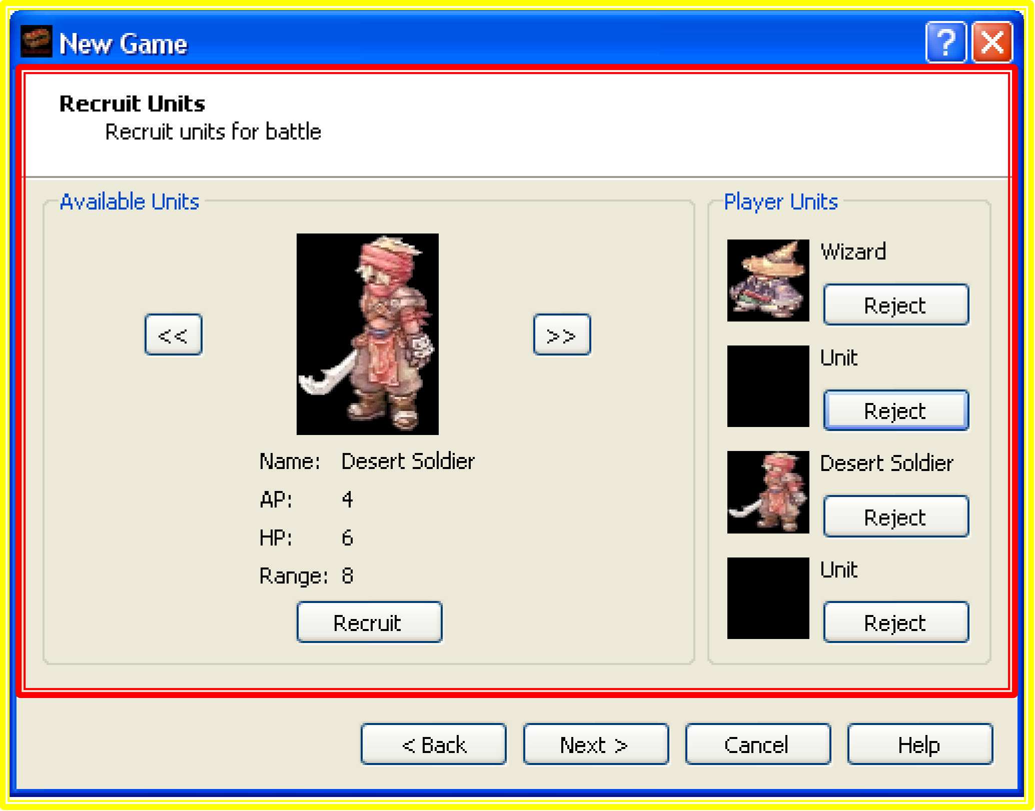
Implementation: Qt Classes

- QMainWindow
- QPushButton
- QCheckBox
- QDialog
- QWizard
- QWizardPage

User Interface Module

QWizard and QWizardPage

TEAM GOLD



New Game

Recruit Units
Recruit units for battle

Available Units

Name: Desert Soldier
AP: 4
HP: 6
Range: 8

Player Units

Wizard
Unit
Desert Soldier
Unit

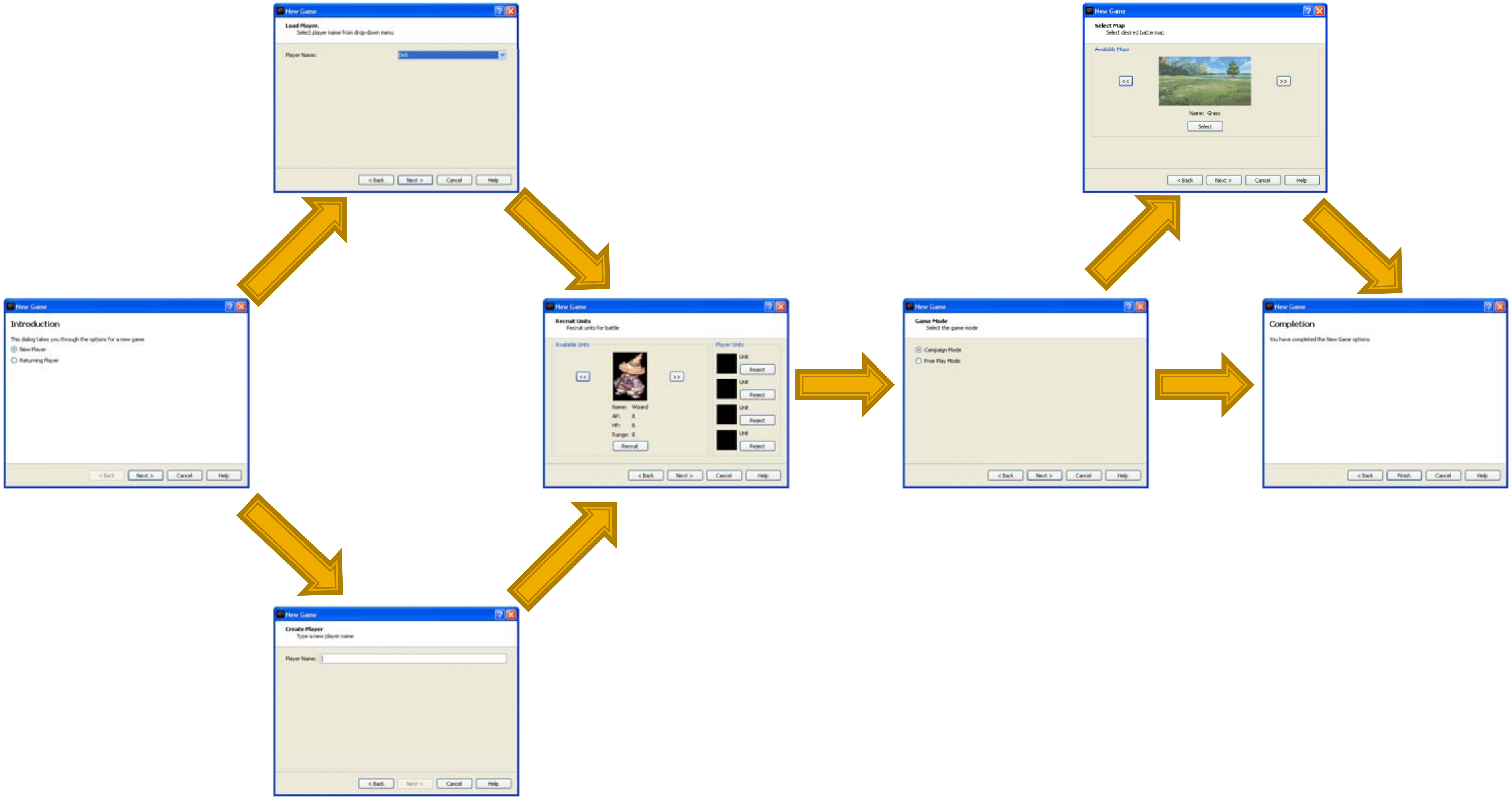
QWizard

QWizardPage

User Interface Module

QWizard & QWizardPage: New Game Dialog Traversal Paths

TEAM GOLD



New Game

Introduction

This dialog takes you through the options for a new game

● New Player

○ Returning Player

New Game

Load Player.
Select player name from drop-down menu

Player Name: Dch

New Game

Create Player
Type a new player name

Player Name:

New Game

Recruit Units
Recruit units for battle

Available Units

Name: Wizard
AP: 8
HP: 8
Range: 8

New Game

Game Mode
Select the game mode

● Campaign Mode

○ Free Play Mode

New Game

Select Map
Select desired battle map

Available Maps

Name: Grass

New Game

Completion

You have completed the New Game options

User Interface Module

Unit Testing & Verification

- Database
  - Generate sprites, maps & test users
  - Perform data reads & writes during new game dialogs
- AI
  - Generate end turn signal
- Mechanics
  - Generate battle over signal

User Interface Module

Challenges

- Object-oriented design knowledge
- Qt knowledge
- Modularity in game development
- Time constraint

User Interface Module

Summary

- Absolutely a fun project
- Relative knowledge of final product
- Code complexity vs. user-friendliness

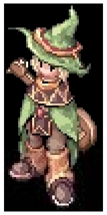
Game Mechanics and AI

Presenter: Josh Kilgore

- Unit
- Player
- Mechanics
  - Move
  - Attack
  - Etc.
- Artificial Intelligence
  - What
  - How

Unit Class

- More than just a pretty face
  - Health
  - Attack Power
  - Attack Range
  - Action Points
  - Movement Rate
  - XP
  - Upgradable Attributes
- Teamwork is Key



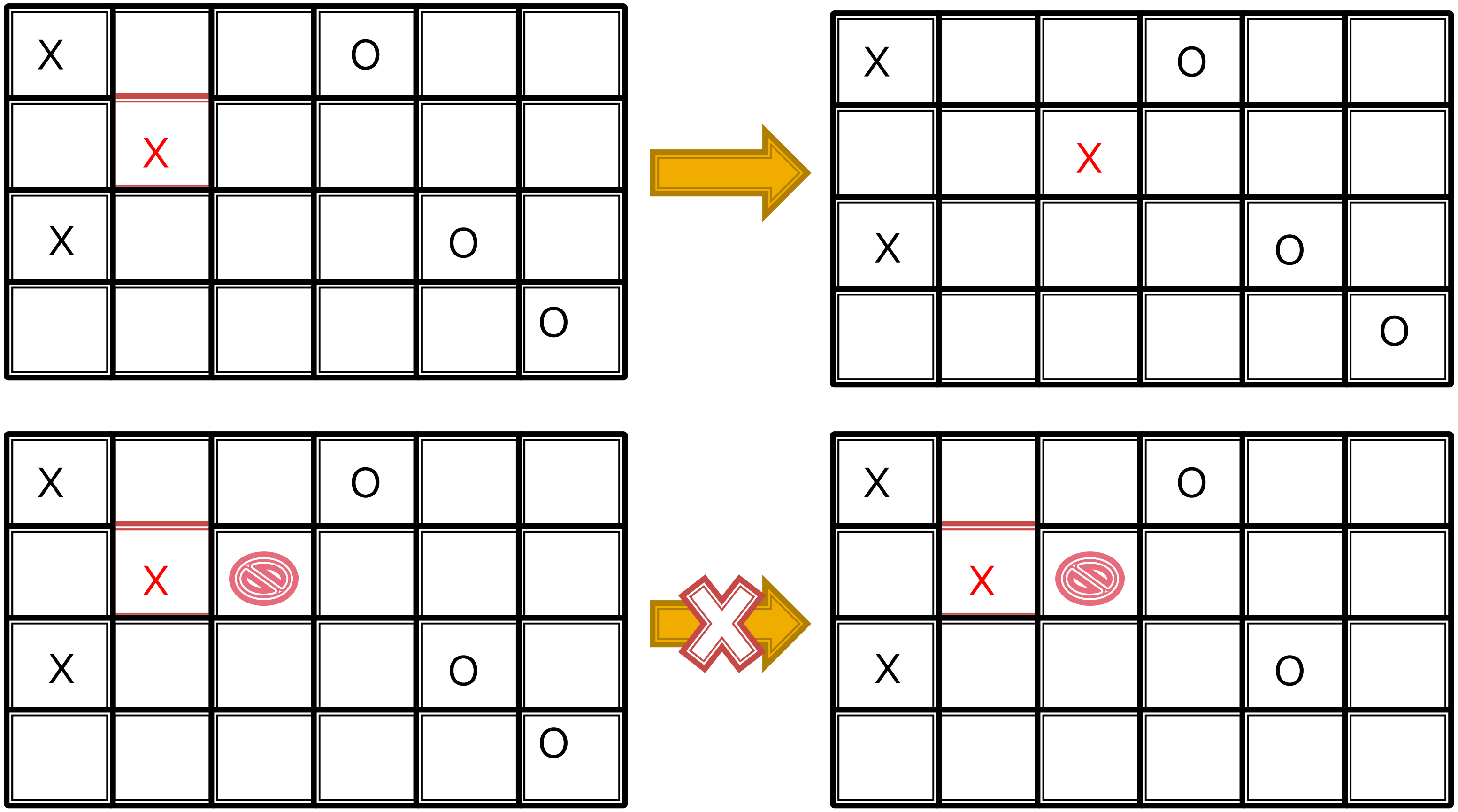
Player Class

TEAM GOLD

- Keep Login ID
  - Keep Upgraded Units
  - Continue Campaign from save point
  - XP Ranking

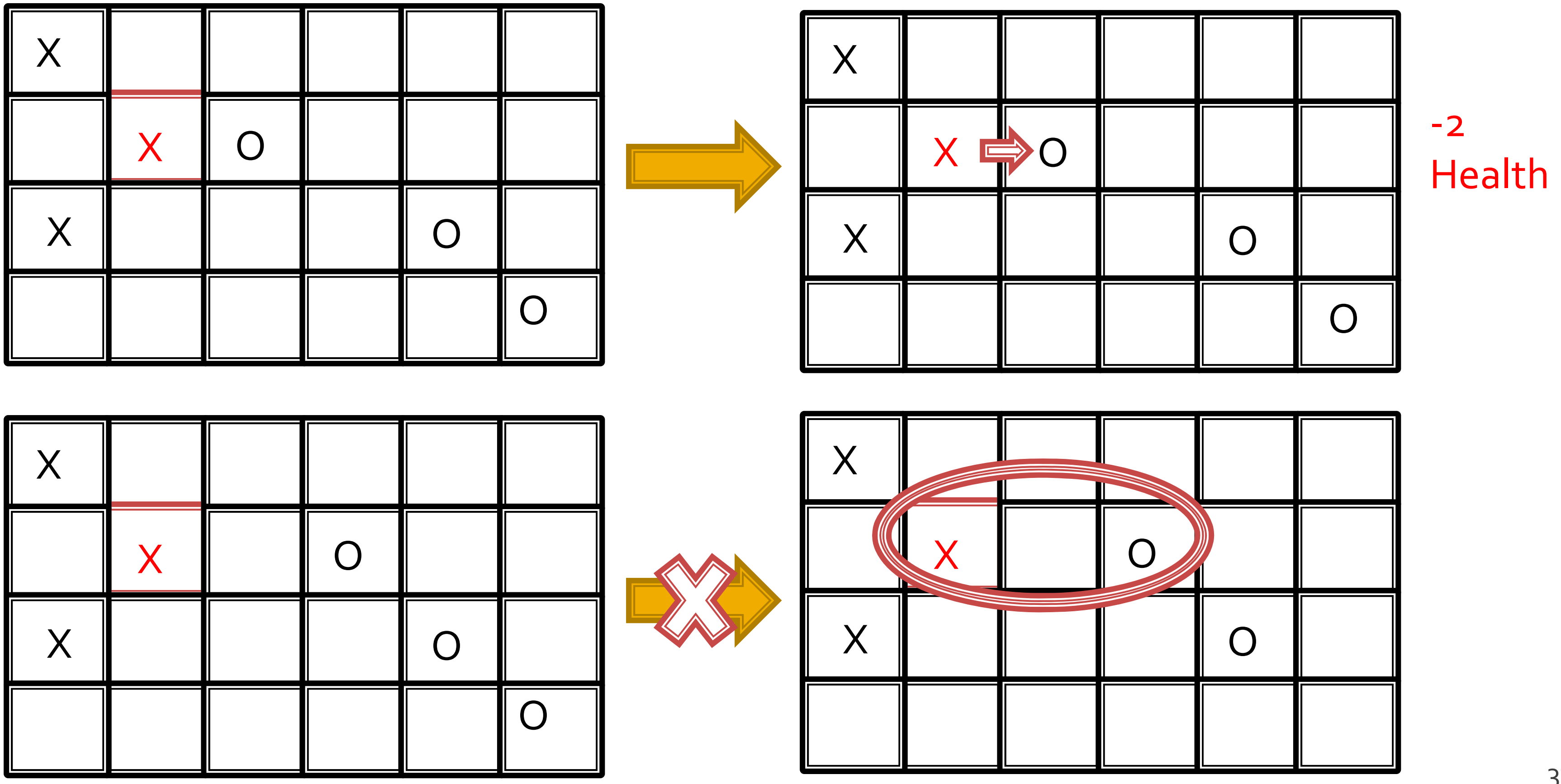
Game Mechanics

- Movement (X moves, cost Action Points)



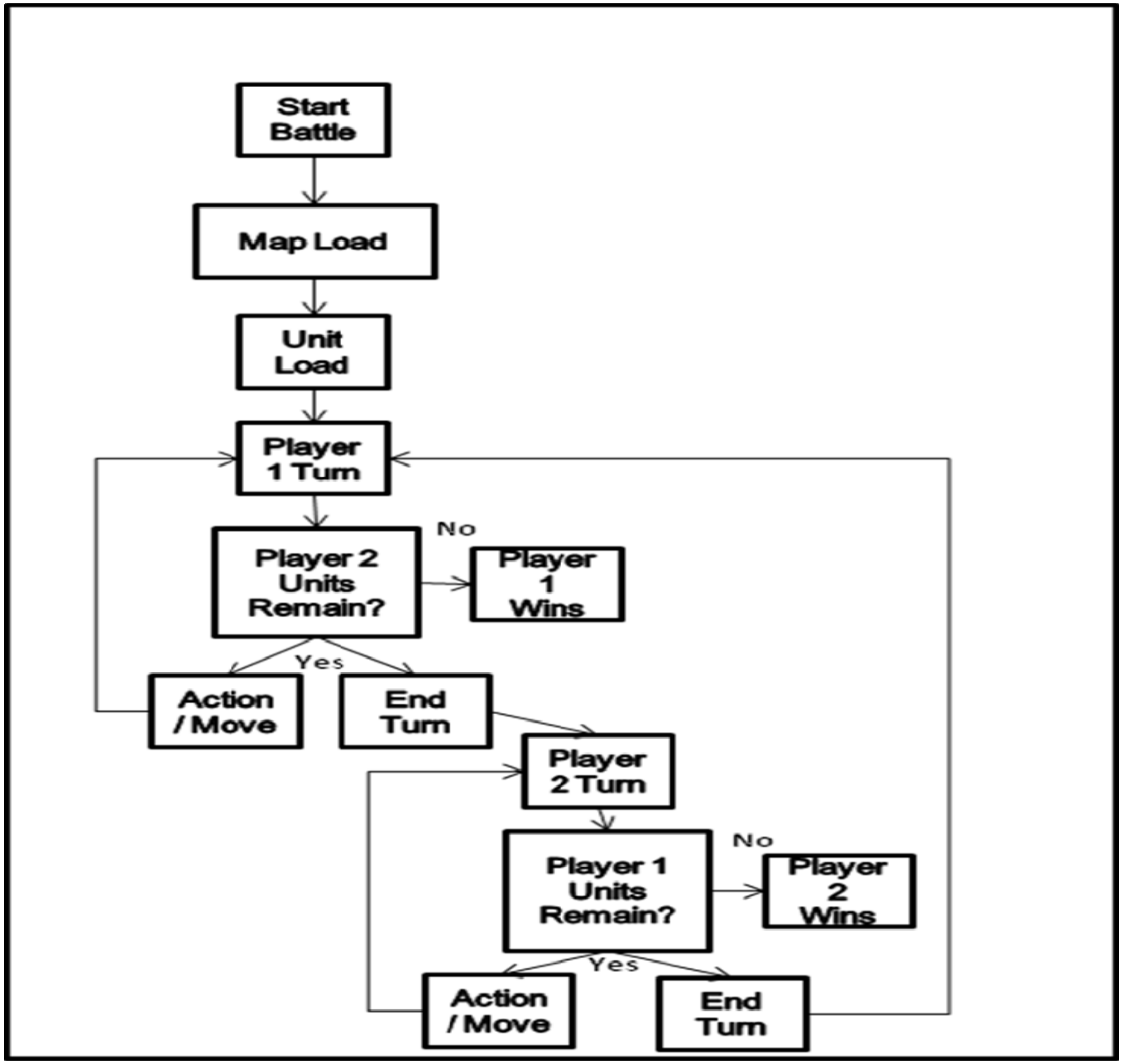
Game Mechanics

- Attack (X attacks O, range = 1)



Game Mechanics

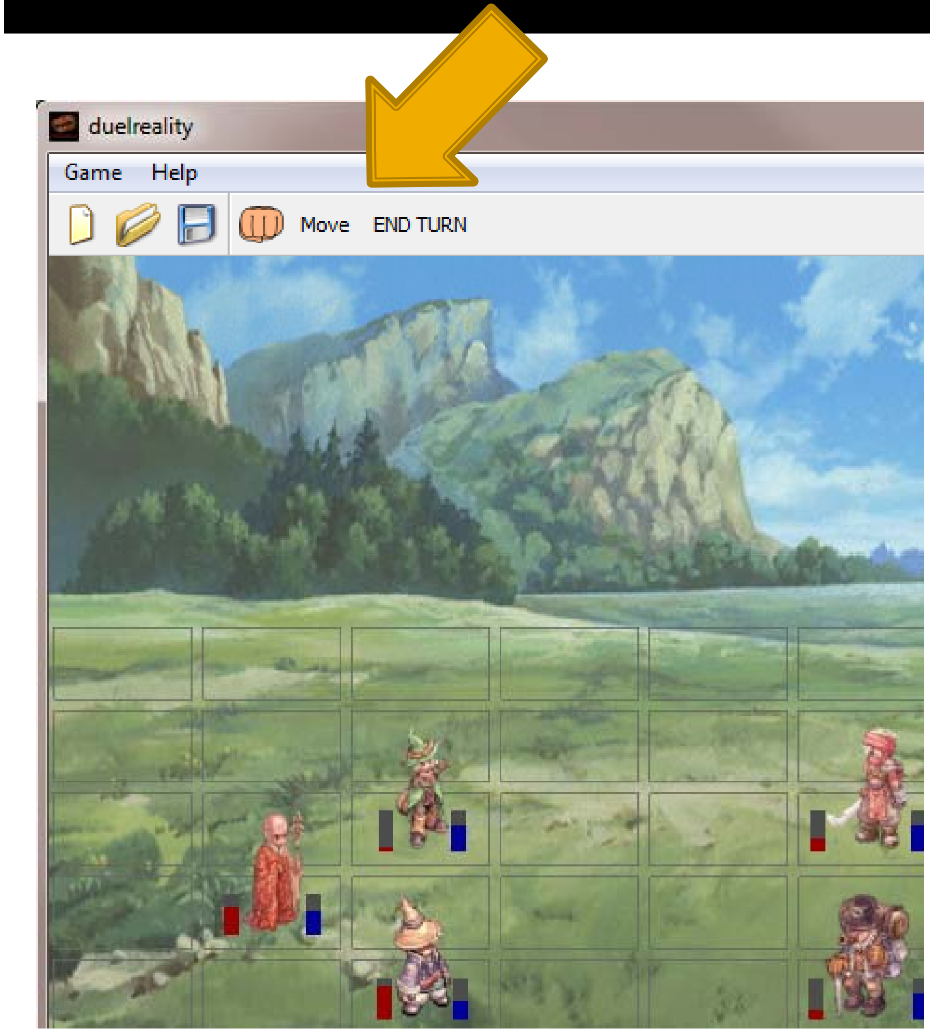
TEAM GOLD



switchPlayers()
isGameOver();
moveComplete();
isValidMove();
isValidAttack();
isOccupied();

Player Interaction

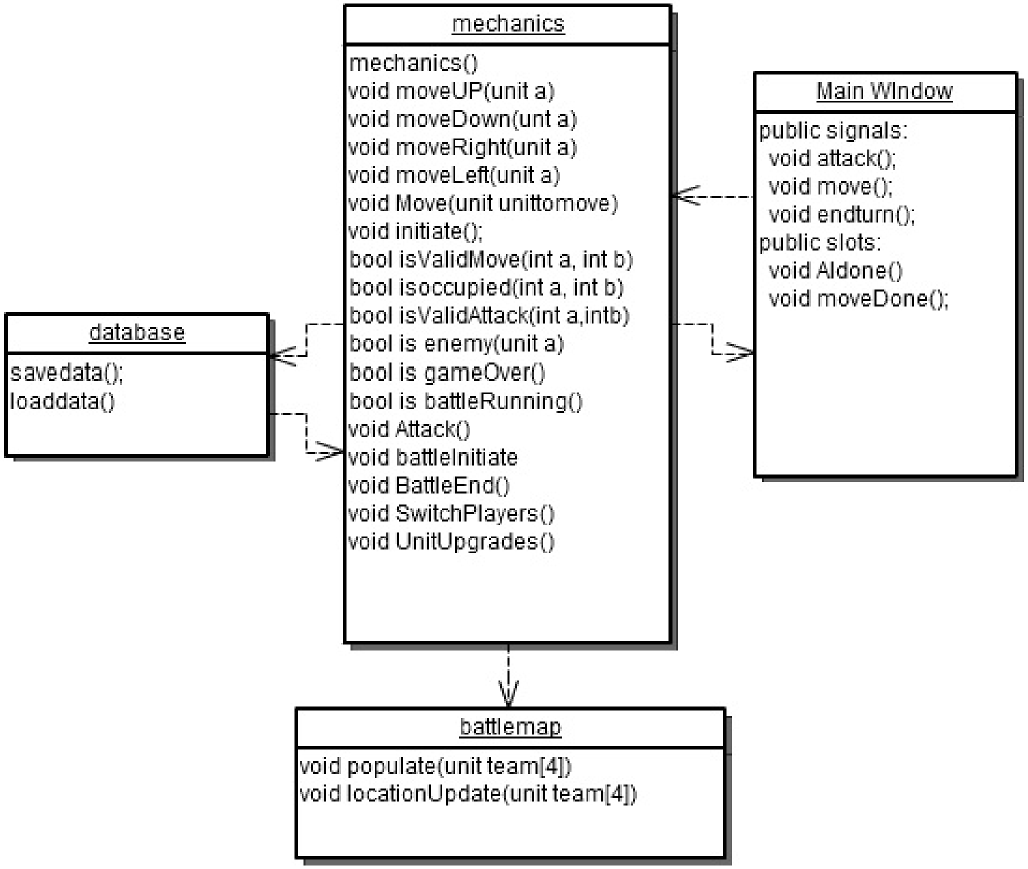
TEAM GOLD



- Player sees icons
  - Signals & slots

Game Mechanics Class Diagram

TEAM GOLD



mechanics

mechanics()
void moveUP(unit a)
void moveDown(unt a)
void moveRight(unit a)
void moveLeft(unit a)
void Move(unit unittomove)
void initiate();
bool isValidMove(int a, int b)
bool isoccupied(int a, int b)
bool isValidAttack(int a,intb)
bool is enemy(unit a)
bool is gameOver()
bool is battleRunning()
void Attack()
void battleInitiate
void BattleEnd()
void SwitchPlayers()
void UnitUpgrades()

Main WIndow

public signals:
void attack();
void move();
void endturn();
public slots:
void AIdone()
void moveDone();

database

savedata();
loaddata()

battlemap

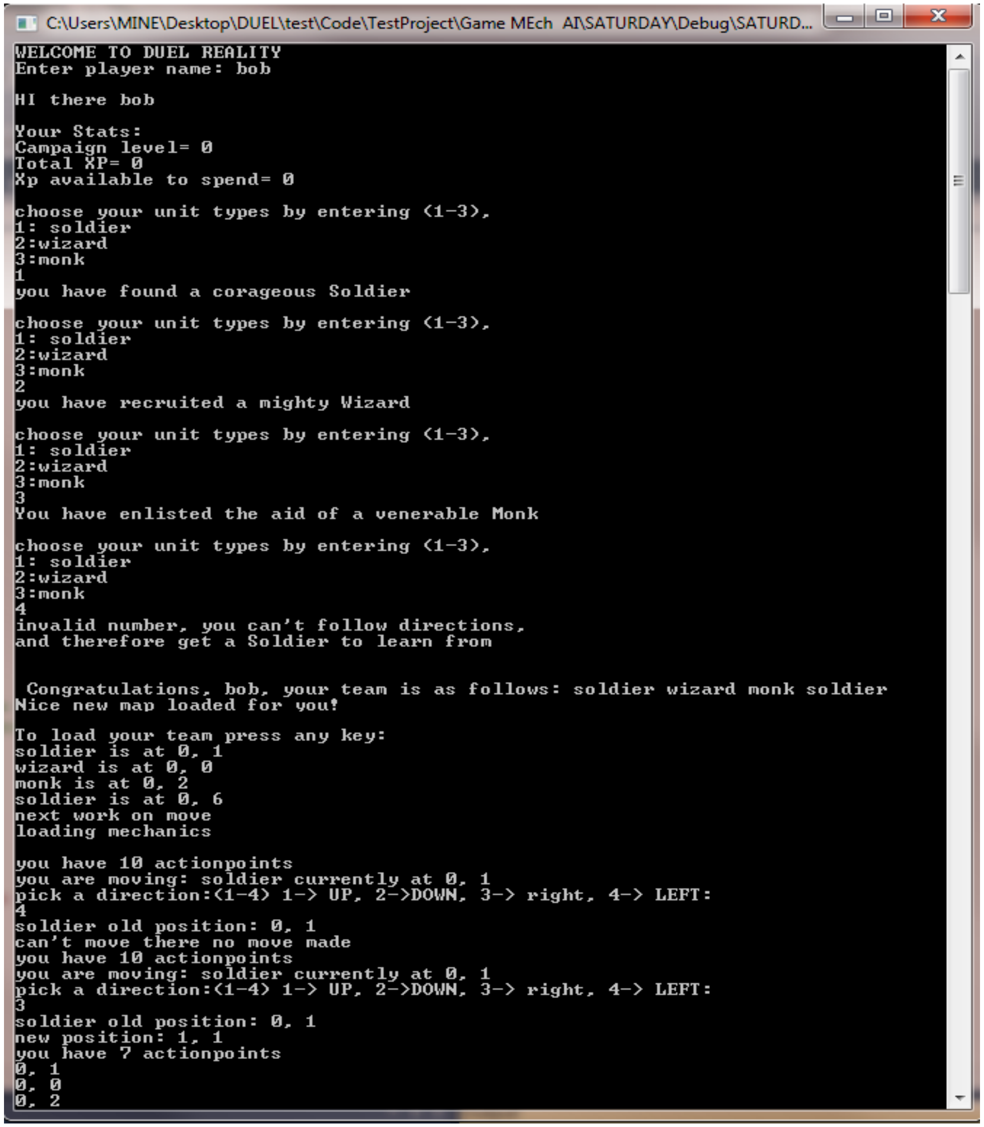
void populate(unit team[4])
void locationUpdate(unit team[4])

- Rely on Database to store unit data
- Internal Manipulation of data

Game Mechanics Verification

TEAM GOLD

- Initial Unit Testing - Console Version
- Module Testing
- System Testing

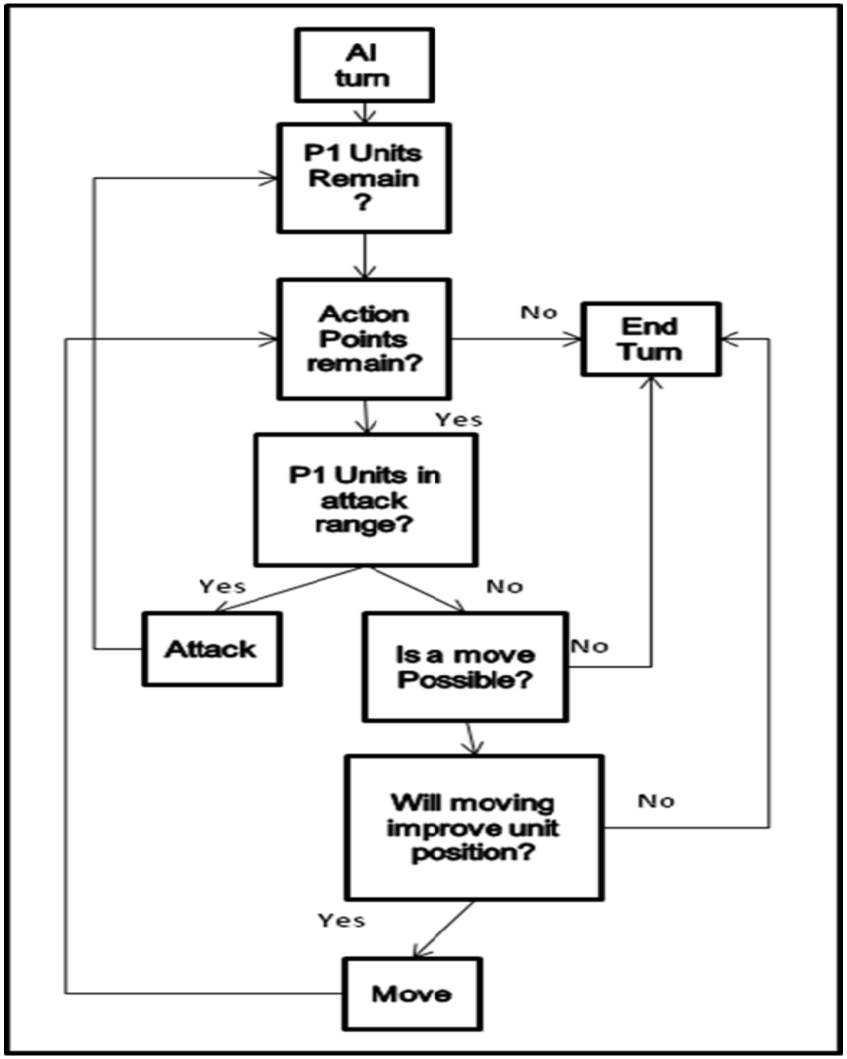

C:\Users\MINE\Desktop\DUEL\test\Code\TestProject\Game MEch AI\SATURDAY\Debug\SATURD...

```
WELCOME TO DUEL REALITY
Enter player name: bob

HI there bob

Your Stats:
Campaign level= 0
Total XP= 0
Xp available to spend= 0

choose your unit types by entering (1-3),
1: soldier
2:wizard
3:monk
1
you have found a corageous Soldier

choose your unit types by entering (1-3),
1: soldier
2:wizard
3:monk
2
you have recruited a mighty Wizard

choose your unit types by entering (1-3),
1: soldier
2:wizard
3:monk
3
You have enlisted the aid of a venerable Monk

choose your unit types by entering (1-3),
1: soldier
2:wizard
3:monk
4
invalid number, you can't follow directions,
and therefore get a Soldier to learn from


 Congratulations, bob, your team is as follows: soldier wizard monk soldier
Nice new map loaded for you!

To load your team press any key:
soldier is at 0, 1
wizard is at 0, 0
monk is at 0, 2
soldier is at 0, 6
next work on move
loading mechanics

you have 10 actionpoints
you are moving: soldier currently at 0, 1
pick a direction:(1-4) 1-> UP, 2->DOWN, 3-> right, 4-> LEFT:
4
soldier old position: 0, 1
can't move there no move made
you have 10 actionpoints
you are moving: soldier currently at 0, 1
pick a direction:(1-4) 1-> UP, 2->DOWN, 3-> right, 4-> LEFT:
3
soldier old position: 0, 1
new position: 1, 1
you have 7 actionpoints
0, 1
0, 0
0, 2
```

AI

- Provide Player with playable opponent
- Decision Tree
  - Board Evaluation
  - Decide on Best Action
- Same constraints as Player actions

AI - Decision Tree



AI turn

P1 Units Remain ?

Action Points remain?

No

End Turn

Yes

P1 Units in attack range?

Yes

No

Attack

Is a move Possible?

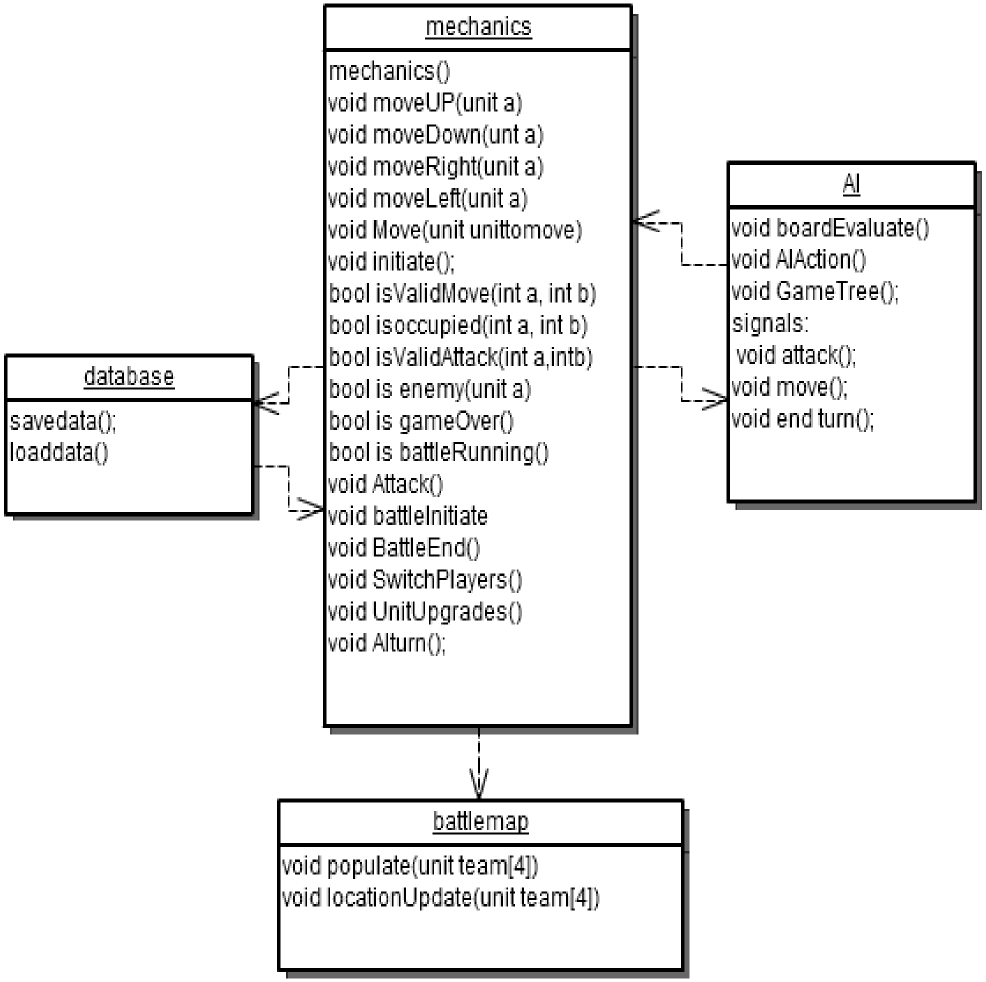
No

Will moving improve unit position?

No

Yes

Move



mechanics

mechanics()
void moveUP(unit a)
void moveDown(unt a)
void moveRight(unit a)
void moveLeft(unit a)
void Move(unit unittomove)
void initiate();
bool isValidMove(int a, int b)
bool isoccupied(int a, int b)
bool isValidAttack(int a,intb)
bool is enemy(unit a)
bool is gameOver()
bool is battleRunning()
void Attack()
void battleInitiate
void BattleEnd()
void SwitchPlayers()
void UnitUpgrades()
void AIturn();

AI

void boardEvaluate()
void AIAction()
void GameTree();
signals:
void attack();
void move();
void end turn();

database

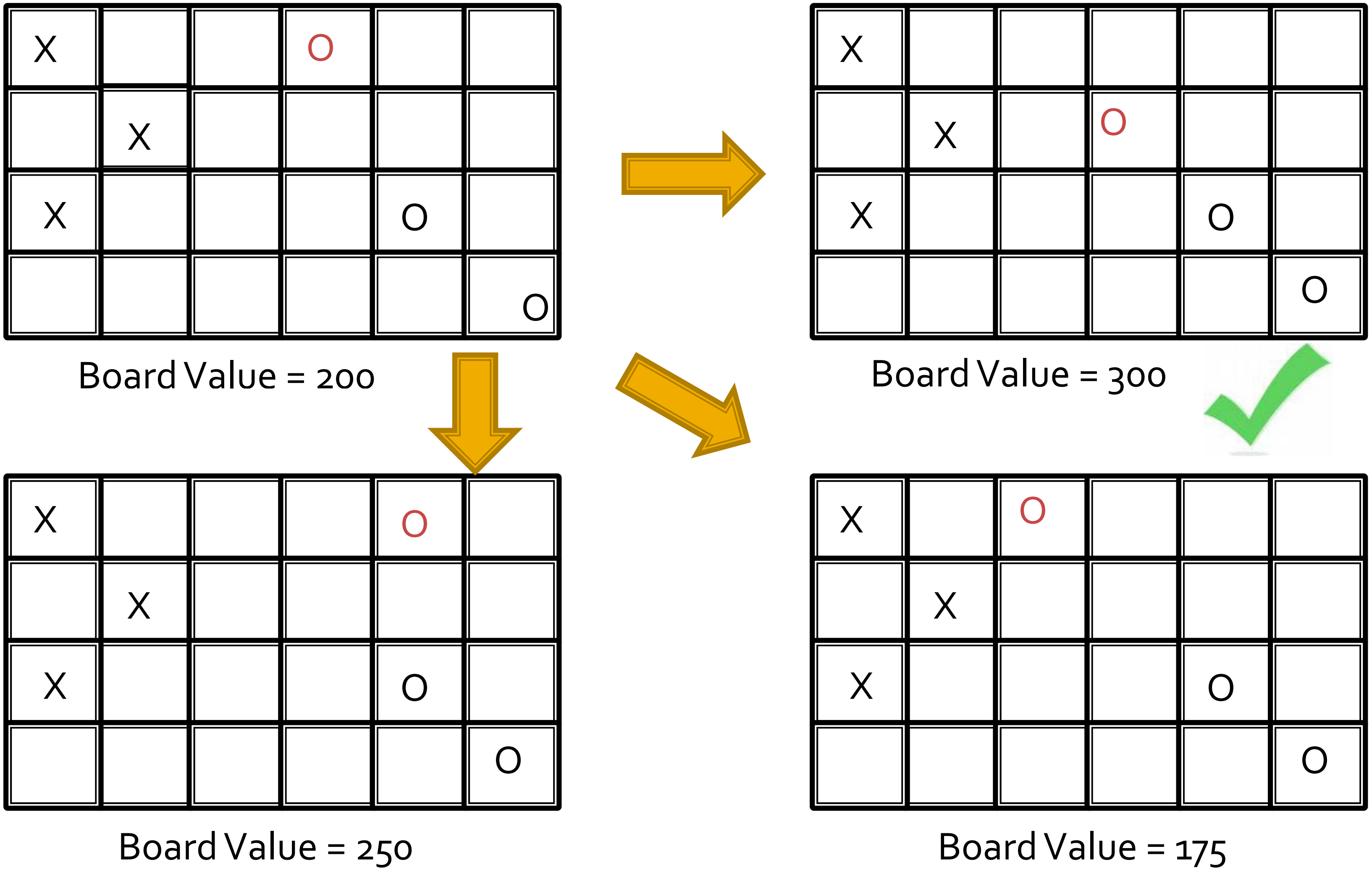
savedata();
loaddata()

battlemap

void populate(unit team[4])
void locationUpdate(unit team[4])

AI

TEAM GOLD



X			O		
	X				
X				O	
					O

Board Value = 200

X					
	X		O		
X				O	
					O

Board Value = 300

X				O	
	X				
X				O	
					O

Board Value = 250

X		O			
	X				
X				O	
					O

Board Value = 175

Game Mechanics & AI Conclusion

- Game Mechanics
  - Classes
  - Functions
- AI
  - Opponent generation

Database

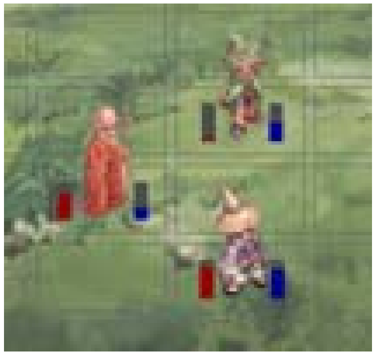
Presenter: Ye Tian

- Database in the game
  - Software-based containers
  - Storage and retrieval
- Database Design
  - SQLite
  - Qt classes
- Database Classes
  - Database
  - Test window
- Summary

Database in the game

- Software-based containers

Game front

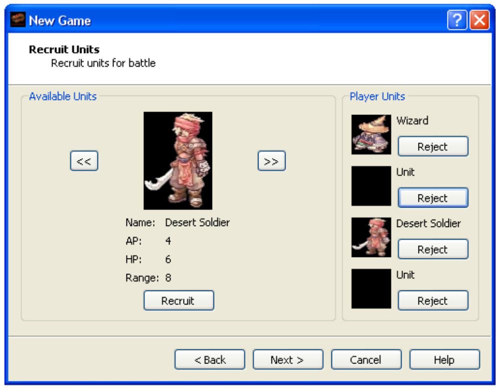


Game background

Unit	Attack	Health	Experience	Location
1				
2				
3				

Database in the game

- Storage and retrieval



New Game

Recruit Units

Recruit units for battle

Available Units

<< >>

Name: Desert Soldier

AP: 4

HP: 6

Range: 8

Recruit

Player Units

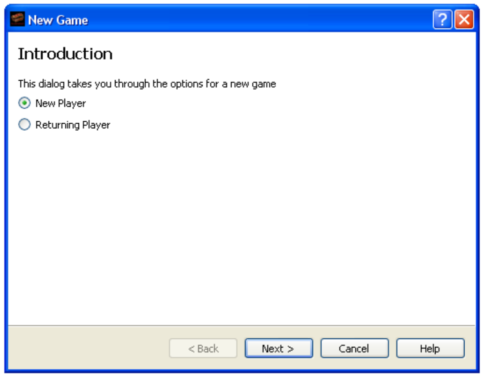
Wizard Reject

Unit Reject

Desert Soldier Reject

Unit Reject

< Back | Next > | Cancel | Help



New Game

Introduction

This dialog takes you through the options for a new game

● New Player

○ Returning Player

< Back | Next > | Cancel | Help

Retrieval

Storage

Unit	Attack	Health	Experience	Location
1				
2				
3				

ID	Player
1	uml
2	dog
3	sleepwalker
4	nickname

Database Design

TEAM GOLD

SQLite

Self-contained
Serverless
Zero-configuration
Transactional
Public domain

Database Design

TEAM GOLD

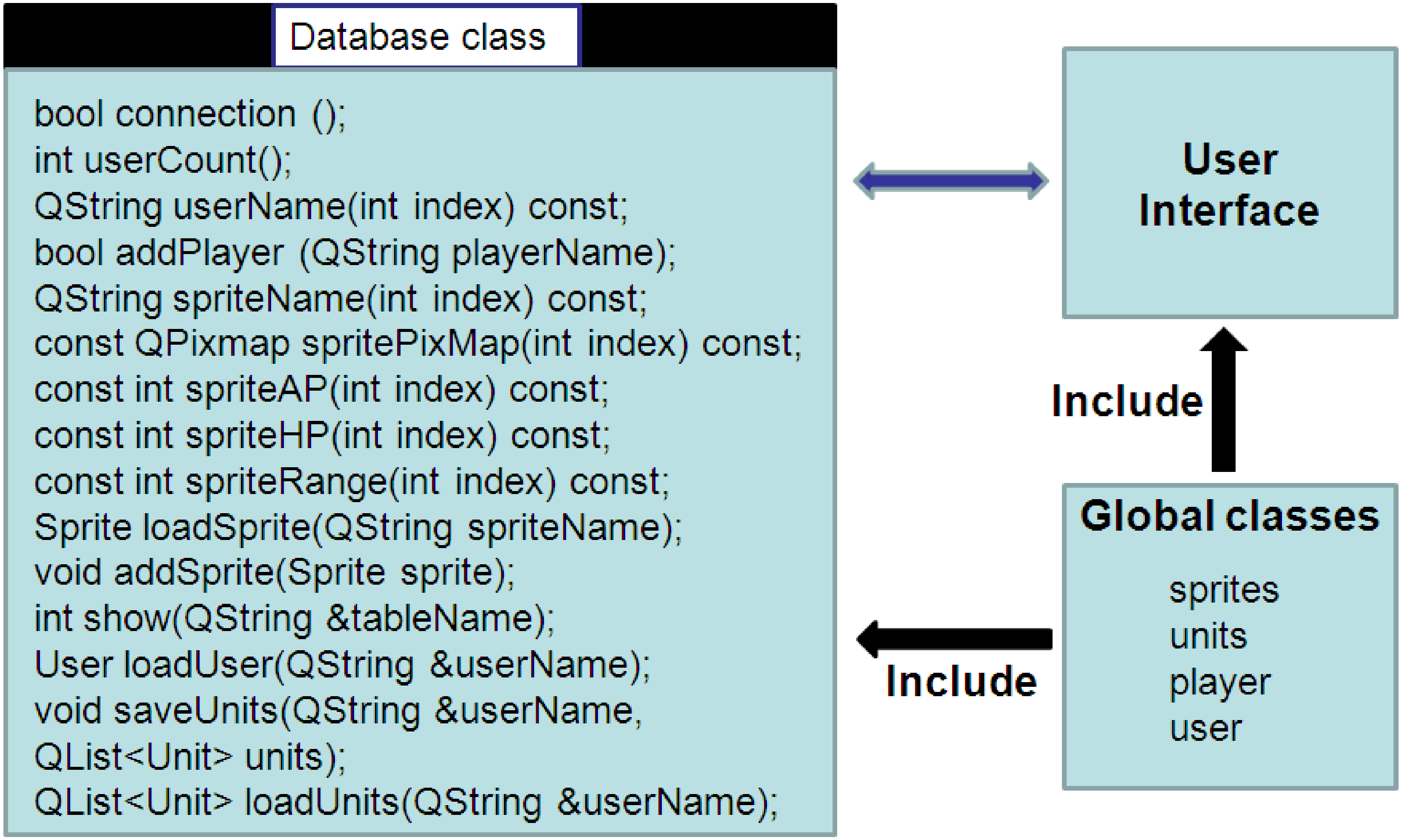
Qt classes

QSqlDriver
QSqlDatabase
QSqlQuery
QSqlTableModel
QSqlRelationalTableModel
QSqlRecord
QSqlIndex
QSqlField
QSqlError

Database Classes

TEAM GOLD

- Database



Database class

```
bool connection ();
int userCount();
QString userName(int index) const;
bool addPlayer (QString playerName);
QString spriteName(int index) const;
const QPixmap spritePixMap(int index) const;
const int spriteAP(int index) const;
const int spriteHP(int index) const;
const int spriteRange(int index) const;
Sprite loadSprite(QString spriteName);
void addSprite(Sprite sprite);
int show(QString &tableName);
User loadUser(QString &userName);
void saveUnits(QString &userName,
QList<Unit> units);
QList<Unit> loadUnits(QString &userName);
```

User Interface

Include

Global classes

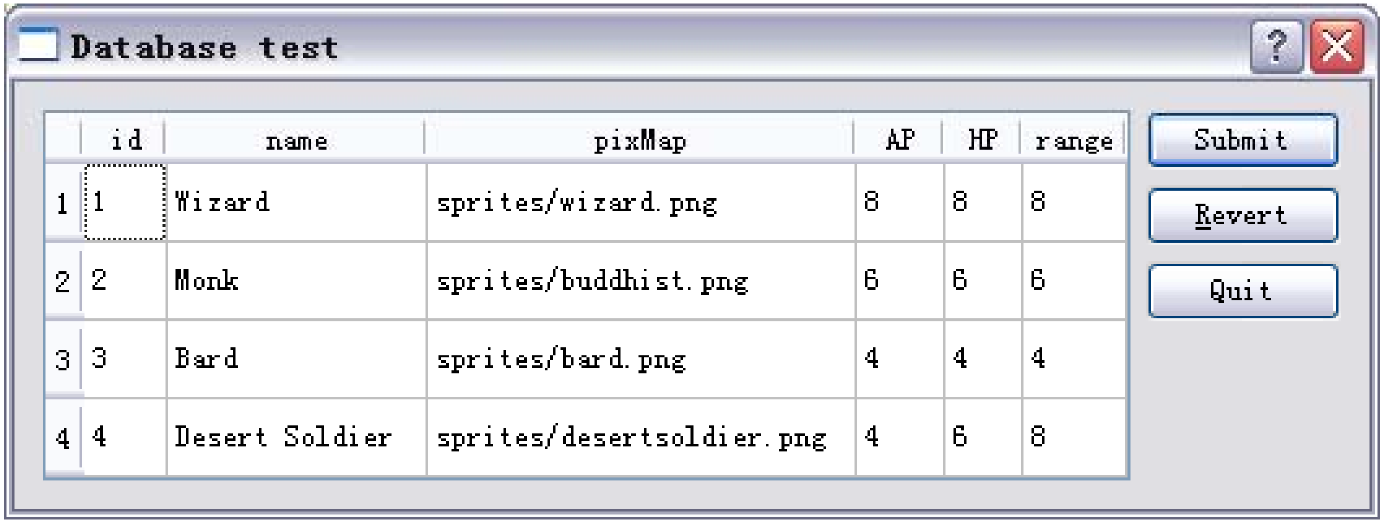
sprites
units
player
user

Include

Database Classes

- Test window

```
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   Database gamedata;
   gamedata.connection();
   gamedata.addPlayer(“sprites”);
   gamedata.show("sprites");
}
```



Database test

	id	name	pixMap	AP	HP	range
1	1	Wizard	sprites/wizard.png	8	8	8
2	2	Monk	sprites/buddhist.png	6	6	6
3	3	Bard	sprites/bard.png	4	4	4
4	4	Desert Soldier	sprites/desertsoldier.png	4	6	8

Database Classes

- After unit test

Database module will be added into the entire project through following three steps:

1. Header file and source file

   database.h; database.cpp

2. SQLITE database file

   gamedata.db3

3. Project file

   QT += sql;

Summary

TEAM GOLD

- Good organization for the game data.
- Useful for Storage and for game.
- Interesting but challengable.

Duel Reality: Conclusion

- Told you about our awesome game
  - Details
  - Modules – Graphics, UI, Game Mechanics &AI, Database
- Pre-orders available
- Thanks for your attention

Questions?

TEAM GOLD