Computer Games Development I
Summer 2009
Project Goals

- Get a basic understanding of the game development process
- Extend or create a 2D platform game
- Work as a team
Project Group - Roles

- Project Manager / Team Leader
  - Organize the group
  - Meetings
  - Responsible for maintaining time reports
    - Members should submit time reports to PM

- Other roles – not necessarily one-to-one
  - Level designer
  - Graphics
  - Game play / story
  - Coder
Project Group

- Group friction?
  - Remember, it is a **team effort**
    - Avoid “I did my part so I’ll just sit by and watch” - Attitude
  - If you have any problems with the group, contact us **immediately** and we will try to solve them

- Other problems?
  - Contact Simon or Justin, 
    
    gamescoursestaff@it.uu.se
Before Coding: Planning is Important

■ What sort game do you want to make?
  • Strategy, Adventure Game, RPG, Action, Shoot'em-up, etc.

■ List desired features of the game
  • Multi-player or Single-player
  • AI
  • Movements and Interaction
  • Physics
  • Sound / Music, etc.

■ Milestones and Time Schedule
Planning: Simple Game Concept

- Introduction
  - Short, exciting, contains important information
- Background (optional)
- Description
  - Player’s experience, key game play elements
- Key features
  - List of most important and outstanding features
- Genre
- Platform(s)
- Concept art (optional)

http://www.gamasutra.com/features/19991019/ryan_02.htm
Planning: Story Board

A (simple) story board can be very useful (cut-scenes / story based games, i.e. Adventure games)

- Water thrashing
  Man suddenly awoken by tug on line

- Water thrashing, grunts
  Man pulls back on line, struggles

- Silence
  Pulling on fishing rod stops

- Large splash, gasps
  Huge fish jumps out of water and consumes man
Three Meetings

- Action plan: Week 26
- Prototype: Week 31
- Examination: Week 35
- The whole group should participate in all meetings
- During the week before each meeting the PM should:
  - Make an appointment with Simon
  - Submit the requested documents
    - Action Plan
    - Progress Report / Updated Action Plan
    - Final Report
First Meeting: Action plan

- Week 26
- The entire group meets and we discuss your Action Plan

Action Plan:
- Simple Game Concept
- Milestones and Time Schedule
  - With prioritized list of features
- Work division, who is primarily responsible for what
Progress Meeting: Prototype

- **Week 31**
- The entire group meets and you demonstrate a working prototype of your game
- PM reports on work division and time spent
- **Progress Report:**
  - In order to get a working prototype you probably have to leave features out or provide limited functionality, describe differences from your original proposal
  - Describe any problems you struggled with and (hopefully) the solutions you came up with
- **Updated Action Plan**
Final Meeting: Examination

- Week 35
- Demo your game
- Answer questions about implementation
- All members must attend

Final Report:
- Differences, Final Game ↔ Proposal
- Which problems arose, (how) could you solve them?
- How could your game be improved (if you had more time)?
- What would you do different, if you would start all over?
- Which lessons have you learned?

Account for work division and time spent
Project Tools

- CVS – code version management
  - TortoiseCVS (windows)
    - Graphical interface integrated into Explorer
  - Unix/cygwin - Command line CVS

- ANT
  - Build tool

- Netbeans (java IDE)
  - Supports ANT and CVS
CVS - Concurrent Versions System

- Handles multiple users working on the same file(s)
- Merges files
- Allows getting different versions
- User typically get latest versions make changes and commits them, making them available to other users

- Important to commit only working code
CVS – cygwin / unix

- Remote and shared repository
- Most used commands (cygwin / unix)
  - First time get files by `cvs checkout game`
  - To get latest changes use `cvs update`
  - Submit changes to files by `cvs commit`
  - Add new files `cvs add *.java` then commit to make it happen
  - Always update before commit
- Must have a correct CVSROOT
  - `:pserver:spel09_NN:pass@cvs.srv.it.uu.se:/spel09_NN`
    or
  - `:pserver:spel09_NN@cvs.srv.it.uu.se:/spel09_NN`
  - Module to checkout: `game`
  - Passwords will be distributed to each group in the beginning of next week
TortoiseCVS - windows

- Right click to access menu
TortoiseCVS

- Adding files, committing changes

A new file was created, add it to the repository.

doc.txt was modified, commit to the repository.

Note the difference in icons.
TortoiseCVS

- Get updated files from the repository
  - Changes will be automatically merged
  - Conflicts are handled manually.
CVS – Resolve Conflict

hej.txt:  User 1

Hej
Wrld

Hej
World

User 2

hej.txt:

changes hej.txt

cvs commit

changes hej.txt

cvs commit

Error: “not up to date”

resolves conflict

cvs commit
ANT – simple example

- Assume we have the following code
  - In ./src/mypkg/HelloWorld.java

```java
package mypkg;

public class HelloWorld {
    public void main(String[] a) {
        System.out.println("Hello World!");
    }
}
```
ANT – simple example

./build.xml:

```xml
<project>
  <target name="clean">
    <delete dir="build"/>
  </target>

  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes"/>
  </target>

  <target name="jar">
    <mkdir dir="build/jar"/>
    <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
      <manifest>
        <attribute name="Main-Class" value="mypkg.HelloWorld"/>
      </manifest>
    </jar>
  </target>

  <target name="run">
    <java jar="build/jar/HelloWorld.jar" fork="true"/>
  </target>

  <target name="all" depends="compile, jar"/>
</project>
```
ANT – simple example

- Using ant
  - ant clean
    - removes the build directory

... in the build.xml:

```xml
<target name="clean">
  <delete dir="build"/>
</target>
```
ANT – simple example

- Using ant
  - ant compile
    - Compiles HelloWorld.java and place the class-file in ./build/classes/mypkg

... in the build.xml:

```xml
<target name="compile">
  <mkdir dir="build/classes"/>
  <javac srcdir="src" destdir="build/classes"/>
</target>
```
ANT – simple example

■ Using ant

• ant jar
  ▪ creates a jar file with mail class HelloWorld.jar
  ▪ can be run with java -jar HelloWorld.jar

... in the build.xml:

```xml
<target name="jar">
  <mkdir dir="build/jar"/>
  <jar destfile="build/jar/HelloWorld.jar"
      basedir="build/classes">
    <manifest>
      <attribute name="Main-Class"
          value="mypkg.HelloWorld"/>
    </manifest>
  </jar>
</target>
```
ANT – simple example

- Using ant
  - ant run
    - Run the created jar file

... in the build.xml:

```xml
<target name="run">
  <java jar="build/jar/HelloWorld.jar" fork="true"/>
</target>
```
ANT – simple example

- Using ant
  - ant all
    - Dependencies on compile and jar

... in the build.xml:

```xml
<target name="all" depends="compile, jar" />
```
ANT – download examples

Two examples, download from course page.

- Example1:
  - the one just showed

- Example2:
  - A simple ant jar example, with java code that loads an image from within a jarfile and shows it in a window
Other Tools

- **IDE - Integrated Development Environment**
  - Netbeans (http://www.netbeans.org/)
  - Eclipse (http://www.eclipse.org)
  - IntelliJ IDEA

- **ANT** (http://ant.apache.org/)
  - Some IDEs have automatic generation of ant build scripts
  - Can create jar files for you, very convenient

- **Graphics**
  - Inkscape – vector graphics drawing program
  - Paint.NET – image editor
  - The GIMP
  - Artweaver