Project – Games Development

Jim Holmström
Project Description

- **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
  - 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.
    - “I did my part so I’ll just sit by and watch”
  - If you have any problems with the group, contact us immediately and we will try to solve them.

- Other problems?
  - Contact Jim or Justin,
    gamescoursestaff@list.it.uu.se
Planning is important

- Before coding…
  - What sort game do you want to make
    - Strategy, build and destroy, adventure, action/shoot'em-up, etc.
  - List desired features of the game
    - Multi-play or Single play
    - AI
    - Movements and Interaction
    - Physics
    - Sound/music, etc…
  - Milestones and time plan
First Meeting – Action plan

- Week 25  (Thursday and Friday)
  - PM makes an appointment with Jim.
  - The entire group meets and we discuss your action plan, that the group has prepared.
  - The Action plan contains,
    - Description of the game idea
    - Story-board / feature list
    - Milestones and time schedule
      - With prioritized list of features.
    - Work division, who is primarily responsible for what.
Final Meeting – Examination

- W34  (August)  21-23
  - PM book the meeting
  - Demo your game
  - Answer questions about implementation
  - All members must attend
  - 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

- Graphics
  - Inkscape – vector graphics
  - Paint.NET – simple paint program

- ANT
  - Build tool

- Netbeans (java IDE)
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)
  - `cvs <command>`
  - 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.
- Must have a correct CVSROOT
  - `:pserver:spel07_NN:pass@cvs.srv.it.uu.se:/spel07_NN`
  - `:pserver:spel07_NN@cvs.srv.it.uu.se:/spel07_NN`
- Module to checkout: `game`
- Passwords will be distributed to each group.
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

Note the difference in icons.
TortoiseCVS

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

* User 2 – cvs update
* User 1 – cvs update
* User 1 – Changes row 2 of hej.txt to:
  
  "Tjenis"
* User 1 – cvs commit
* User 2 – At the same time, changes row 2 of hej.txt to:
  "hoppla"
* User 2 – cvs update (commit will give error since “not up to date”)
* User 2 – resolves conflict

```
hej.txt:
hejsan
<<<<<<<<< hej.txt
hoppla
========
tjenis
>>>>>> 1.4
```

```
hej.txt:
hejsan
hoppla
```

choose

```
hej.txt:
hejsan
hoppla
```

* User 2 – cvs commit
Tools

- Other tools
  - IDE - Integrated Development Environment
    - Netbeans http://www.netbeans.org/
    - Eclipse http://www.eclipse.org
    - IntelliJ IDEA
  - Inkscape – vector graphics drawing program
  - Paint.NET – image editor
  - ANT
    - Some IDEs have automatic generation of ant build scripts.
    - Can create jar files for you, very convenient
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 I 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.
Deadlines

Project Plan Deadline
21/6, next week.

Start working...