Computer games development I & II

12 June 2008

Project Information, CVS and Ant Java

Karl.marklund@it.uu.se
Very few game books are aimed at providing a basic foundation for creating games. This book tells you what you need to know to start creating games in Java.

Doug Twilleager

Chief Architect for Java Game Technologies
Sun Microsystems
What is this project about?

Get a basic understanding of the game development process.

Extend or create a 2D platform game.

Work as a team!
Roles within the project group

- **Project Manager (PM):**
  - Organize the group
  - Responsible for Meetings
  - Responsible for maintaining time reports
  - Group Members should submit time reports to PM.

- Roles other than PM doesn’t necessarily have to be one-to-one.

**Project Manager**
- Karl

**Level Designer**
- Justin

**Graphics**

**Game play / story**

**Coder**
- Karl

- Roles other than PM doesn’t necessarily have to be one-to-one.
Remember, it is a team effort.

If you have any problems with the group, contact us immediately and we will try to solve them.

Other problems, contact Justin and Karl: gamescoursestaff@list.it.uu.se

Stay away from attitudes similar to “I did my part so I’ll just sit by and watch..”
Before coding: planning is important.

What sort game do you want to make?

Strategy, build and destroy, adventure, action/shoot'em-up, etc…

Milestones and time plan

List desired features of the game:

- Multi-play or Single play
- Artificial Intelligence (AI)
- Movements and Interaction
- Physics
- Sound/music, etc…
A (simple) story board can be very useful.
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
1st meeting: action plan

PM makes an appointment with Justin or Karl by sending an email to gamescoursestaff@list.it.uu.se

The whole group meets and we discuss your action plan that your group has prepared.

Action Plan:
Description of the game idea
Story-board / Feature list

Action Plan (continued):
Milestones and time Schedule.
A prioritiezed list of features (essentials & xtras)
Work division: who is primarily responsible for what.

Before the meeting, send us your action plan document (should be about 2 A4 pages).
2nd meeting week 32: prototype demonstration

Before the meeting, send us your progress report and the updated action plan.

Progress Report:
To be able to get a working prototype you are likely to have to leave features out or provide limited functionality.

Describe how your prototype differs from your original proposal.

Describe any problems you struggled with and (hopefully) the solutions you came up with.

Updated Action Plan:
Update your action plan to accommodate any changes you made.

PM makes an appointment with Justin or Karl by sending an email to gamescoursestaff@list.it.uu.se

The whole group meets and you demonstrate (a working) prototype of your game. PM report on work division and time spent.
3nd meeting week 35: examination

PM makes an appointment with Justin or Karl by sending an email to gamescoursestaff@list.it.uu.se

The whole group meets and you demonstrate your final version of your game. PM report on work division and time spent.

Final Report:
Describe how your final game differs from your original proposal.

Describe any problems you struggled with and (hopfully) the solutions you came up with.

Future improvements (if anyone would continue your work).

Lessons learned?

What to do different if you where to start fresh again?

Before the meeting, send us your final report.
Concurrent Versions System (CVS)

Allows multiple users working on the same file(s).

Merges files.

Allows getting different versions!

User typically get latest versions (cvs update or cvs checkout), make changes and submit new version back to the repository (cvs commit) making them available to other users.

Important to commit only working code!
A good start is to look at the basic 2D platform from chapter 5 in the book by Brackeen.

Each group will be assigned a CVS repository with a directory structure similar to this game as the initial code base.

Passwords will be distributed to each group.

To access the repository you need to set your CVSROOT:

```
CVSROOT=:pserver:spel08_NN:pwd@cvs.srv.it.uu.se:/spel08_NN
```

Module to checkout: game

```
cvs co game
```
First time – get files by:
  `cvs checkout game`

Get latest changes:
  `cvs update`

Submit changes:
  `cvs commit -m "Desc"`

Add new file to the repository:
  `cvs add *.java`

Then commit to push files to repository.
TortoiseCVS – a graphical front-end for windows integrated with Explorer.
Right-click to access TortoiseCVS menu.
A new file was created, add it to the repository.

doc.txt was modified, commit.
Note the difference in icons.
Get updated files from the repository

Changes will be automatically merged
Conflicts are handled manually.
user 1

cvs update

Changes row 2

cvs commit

user 2

cvs update

Changes row 2

cvs commit ➔ error: “not up to date”

cvs update

cvs commit

Manualy resolve conflict
Apache Ant is a software tool for **automating software build processes**. It is similar to make but is written in the Java language, requires the Java platform, and is best suited to building Java projects.

Because Ant made it **trivial to integrate JUnit tests with the build process**, Ant has made it easy for willing developers to adopt **test-driven development**.

The most immediately noticeable difference between Ant and make is that **Ant uses XML** to describe the build process and its dependencies, whereas make has its Makefile format. By default the XML file is named **build.xml**.

Download and documentation:
http://ant.apache.org
Test to see if ANT is already installed...

...darn, ANT was not installed 😞
Installing Ant

The binary distribution of ANT consists of the following directories:

```
ant
  ---- README, LICENSE, fetch.xml, other text files. // basic information
  ---- bin // contains launcher scripts
  ---- lib // contains Ant jars plus necessary dependencies
  ---- docs // contains documentation
  ---- images // various logos for html documentation
  ---- etc // contains documentation (a must read ;-) )
```

Only the bin and lib directories are necessary when installing ANT. You copy these two directories to any place where you want to use your distribution files there. This directory will be known as ANT_HOME.

Windows 95, Windows 98 & Windows ME Note:

On these systems, the script used to launch Ant will have problems if ANT_HOME is a long filename (i.e., a filename which is not of the format known as "8.3"). This is due to limitations in the OS's handling of the "for" batch-file statement. It is recommended, therefore, that Ant be installed in a short, 8.3 path, such as C:\Ant.

On these systems you will also need to configure more environment space to cater for the environment variables used in the Ant launch script. To do this, you will need to add or

[Read about how to download and install ANT.](http://ant.apache.org/manual/index.html)
Tiresome to set these variables after every computer re-start...

Windows and OS/2

Assume Ant is installed in c:\ant. The following sets up the environment:

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk-1.5.0.05
set PATH=%PATH%;%ANT_HOME\bin
```

Assume Ant is installed in /usr/local/ant. The following sets up the environment:

```
export ANT_HOME=/usr/local/ant
export JAVA_HOME=/usr/local
export PATH=${PATH}:${ANT_HOME}/bin
```

Linux/Unix (csh)

```
setenv ANT_HOME /usr/local
setenv JAVA_HOME /usr/local/jdk
set path=| $path $ANT_HOME/bin
```

Having a symbolic link set up to point to the JVM/J2K version makes updates more seamless.
Should be possible to set these variables from the Control Panel...
Instead I saved these command in a .bat file

... But I couldn’t get it to work.
After every computer restart, all I have to do is to run this script and I get all my ANT settings back again.
To check which variables are set.
Now you can run ANT

But you still need to create a build.xml file.
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!");
    }
}
```
To use ANT we can use the following build file `.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>
```
To use ANT we can use the following build file `./build.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 clean` removes the build directory.
To use ANT we can use the following build file `.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>
```

Compiles HelloWorld.java and place the class-file in `./build/classes/mypkg`
To use ANT we can use the following build file `./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>
```

A JAR file (or Java ARchive) is used for aggregating many files into one. It is generally used to distribute Java classes and associated metadata.

ant jar

Creates a jar file with main class HelloWorld.jar

Can be run with java -jar HelloWorld.jar

To use ANT we can use the following build file `./build.xml`:
To use ANT we can use the following build file ./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>
```

Run the created jar file with:

```
ant run
```
You can download two ANT examples from the course homepage.

The Hello World example from the previous slides:

http://www.it.uu.se/edu/course/homepage/games/st08/proj1/ant_example.zip

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

http://www.it.uu.se/edu/course/homepage/games/st08/proj1/ant_example2.zip
ANT

- Art from apache: [ANT manual]
- ANT tutorials can be found here
- You can also find a lot more ANT examples here
- A simple example with Eclipse
  - Example1 ZIP
- A simple ant-jar example:
  - Example2 ZIP

Other tools

- A simple yet powerful pair: Netbeans
  - It will be available in the coming weeks
  - If you would like to download it: Netbeans
- Inkscape, a free vector graphics editor
  - Go to Inkscape: dow
- Netbeans a free IDE for Java
  - http://www.netbeans.org/
- Eclipse a free IDE for java.
  - http://www.eclipse.org/
The module you want

ANT

- Art from apache: [ANT manual](http://www.apache.org/projects/artifact.html)
- ANT tutorials can be found on [SourceForge](http://sourceforge.net)
- You can also find a lot more information on the [Ant Home](http://ant.apache.org)
- A simple example with HE
  - [Example1 ZIP](http://example1.zip)
- A simple ant jar example
  - [Example2 ZIP](http://example2.zip)

Other tools

- A simple yet powerful pair
  - It will be available in
  - If you would like to draw simple vector graphics
    - Inkscape, a free vector graphics editor
      - Go to Inkscape [download page](http://inkscape.org)
  - Netbeans a free IDE for Java
    - [http://www.netbeans.org/](http://www.netbeans.org/)
  - Eclipse a free IDE for java

---

Last updated: 2008-06-18 14:42:15 Resposible: Justin Browser Web-Content
ANT

- Art from: http://commantra.sourceforge.net
- ANT tutorials can be found here: http://www.antopen.org
- You can also find a lot more info on: http://www.redhat.com/ant
- A simple example with Hello: 
  - Example1 ZIP
- A simple ant jar example: 
  - Example2 ZIP

Other tools

- A simple yet powerful pair: Inkscape
  - It will be available in the lab today.
  - If you would like to download it:
    - Go to Inkscape: http://inkscape.org/
- Netbeans a free IDE for Java: 
  - http://www.netbeans.org/
- Eclipse a free IDE for Java: 
  - http://www.eclipse.org/
Other tools that may be useful…

**Paint.NET:** A simple yet powerful paint program called (freeware).
   It will be available in the PC-LAB.
   If you would like to download it you can find it here: [http://www.getpaint.net/download.html](http://www.getpaint.net/download.html)

**Inkscape**, a free vector graphics editor.
   Go to [http://www.inkscape.org/download.php](http://www.inkscape.org/download.php)

**Netbeans** a free IDE for Java (installed in PC-labs)

**Eclipse** a free IDE for java.
Time to get your team together!