CISC/CMPE320

- Notices:
- Assignment 1 due this Friday at 7pm.
- Jira teams assembled. Let me know who the "team leader" will be and of any more membership changes.
- Tomorrow's lecture will be virtual only – no "in person" lecture.

Today

- First team meeting.
- Introduction to "Being Agile".
- Back to C++:
  - Boolean Expressions.
  - Conditionals.
  - Loops.

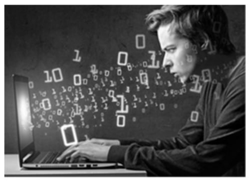
First Team Meeting

- Introduce each other - Exchange contact info?
- Present your background/experience. Which hardware/OS do you like to use for C++ development?
- Has each member read over and understood the projects page? Do you know what the deliverables are and when they are due?
- Pick a notetaker for this meeting.
- Notetaker takes attendance and records any momentous decisions – is anyone missing? Use Confluence to record notes and link them to an issue in Jira. Or, keep separate notes on paper or on a laptop and copy them into Confluence later.

First Team Meeting, Cont.

- Discuss project ideas – negotiate?
- Other meeting time(s) than just the tutorial time?
- If you do settle on a project idea, discuss it with your TA (your mentor) to see if it is too hard/too easy and just what might be involved with implementation. Write an "executive summary" to be sent to the CEO.
- Start thinking about two major roles that must be filled soon: Team Manager (or "Lead") and Primary Software Architect.
- Start putting in your story line so you can build your first sprint!

The Role of the Primary Software Architect



- Forms and owns the mental model of the software product.
  - Includes the detailed specifications for all functionality.
- Must communicate this model to the rest of the team.
- Prevents deviations from the model.
- Must represent the user when it comes time to form tradeoffs between time to completion, cost, performance and ease of use.
- Like a movie director!

Primary Software Architect, Cont.

- Does not necessarily write any code (*in the "real world"*) – in this course he or she will have to!
- This job can be too demanding for one person on a very large project.
- So, it can be split up along very clear system boundaries.
- Object oriented programming languages help enable this splitting up.
- But all architects must agree on the overall vision.

Architect – Builder Interaction

- The code writers or "Builders" implement the architect's vision.
- The architect should not decide on how the functionality is implemented – he can only suggest. This allows the builders to come up with better ideas.
- In CISC/CMPE320 – all team members will be code writers.

Project Management Role

- Or "Team Lead".
- The primary architect will be too busy to fill a management role too.
- So another person should be the project manager.
- Like a movie producer.
- He or she is responsible for the implementation of the architect's vision.

Lots of Other Roles

- Chief Programmer
- Other Programmers, like component programmers and other specialist programmers
- Note-takers
- Archivists (Wiki maintenance)
- Technical Writers (Wiki content)
- Illustrators
- Testers
- Accountant
- Public relations
- Researcher
- Interface specialist
- Artists
- Story writers
- Repository design and maintenance
- …

Agile Teams – Important Characteristics

- Small team, skilled, experienced, able to co-exist.
- Minimal management.
- Strange work hours!
- Coding, design, testing all at the same time.
- Frequent (nightly?) builds.
- Many short efficient meetings (SCRUMs).
- Absolute reliance on CASE tools.
- Rapid prototyping to demonstrate to client/user.
- Frequent input/changes from client/user who may even be a team member.

Agile Teams, Cont.

- Compared to earlier, classical project teams:
  - Smaller.
  - Less formal.
  - Less time spent on documentation.
  - More effective in a shorter time – more people coding.
- But:
  - When an Agile team goes wrong – they go really wrong!
  - Results from a lack of all the formal checks and balances of a more "formal" approach.
  - Difficult to tell when things go wrong. Little accountability. Often have to start again.

Your Team

From this:

To this!:

See:

http://taswar.zeytinsoft.com/2011/01/25/starting-an-agile-team-to-become-the-a-team/

Cute Video

- Being Agile is Our Favourite Thing:

http://www.youtube.com/watch?v=ALWHCUNU8Nw

Boolean Expressions

- We have seen the boolean operators already. Here are a few notes:
- Something like

  `a < b < c`

  will compile and run, but may not produce the desired result. Better to use:

  `a < b && b < c`
- Remember that **&** and **|** are bitwise operators, not logical ones.

Boolean Expressions, Cont.

- The **&&** and **||** logical operators use "short circuit evaluation":
- For **&&** if the LHS is **`false`** then the RHS is not evaluated.
- For **||** if the LHS is **`true`** then the RHS is not evaluated.
- (Same as in Java.)

Boolean Expressions, Cont.

- Non zero integers are treated as being **`true`**, and zero is treated as being **`false`**. (*Ouch*!)
- So, you can use logical operators, **&& ||** and **!**, with integers.
- For example, the code:

```
int x = 10;
if (x)
```

- is the same as:

```
int x = 10;
if (x != 0)
```

Boolean Expressions, Cont.

- Also, this is legal syntax:

```
if (x = 10)
```

- The assignment operator returns the value being assigned, which in this case is a `true`! But suppose `x` is `12` and you really meant to type `==`…
- *Ouch, again*!

Boolean Expressions, Cont.

- See TestStuff.cpp.
- Applying **!** to a non-zero integer returns **`false`** or zero.
- An **`if`** statement will treat a pointer by itself as an integer – it will be **`true`** unless it is **`nullptr`**.
- You can also test assignment statements since the assignment operator returns the value being assigned.

Conditional Expressions

- C++ has if, if/else and switch.
- Syntax of if/else:

```
if (boolean_expression)
   true_part
else
   false_part
```

- Use `{ }` to enclose more than one statement.

`switch` Statement

- Same syntax as in Java:

```
switch (expression) {
   case val1:
      // statements if expression produces val1
      break;
   case val2:
      // statements if expression produces val2
      break;
   case val3:
      …
   default:
      // statements if none of the above is true
} // end switch
```

switch Statement, Cont.

- ***expression*** must be an integer.
- The **`break`** statements are needed to prevent the next case statement from taking place if the one above it is **`true`**.
- The **`default`** is only executed if none of the other **`case`** blocks have executed.

The Conditional Operator

```
int max = (n1 > n2) ? n1 : n2;
```

- Is the same as:

```
int max;
if (n1 > n2)
   max = n1
else
   max = n2
```

- Which is easier to read?

Four Loop Structures:

```
while (condition)
   statement_or_block

do
   statement_or_block
while(condition);

for(init; condition; expr)
   statement_or_block

for(type element : collection) // C++11 only
   statement_or_block
```

Goto

- C++ also has the infamous “goto” statement:

```
goto identifier :

….

identifier : statement
```

- Don’t use it!
- (Maybe use it to kick you completely out of a nested loop. **`break`** only kicks you out one level…)

`break` and `continue`

- Work as they do in Java:
- **`break;`** spits you right out of the loop.
- **`continue;`** sends you back to the loop condition.

string Class

- From the STL.
- Better than C-Strings!
- See StringDemo.cpp
- See CPlusPlus.com for more info and member functions.

vector Class

- Demo of the vector (*a template container class*) class from the STL:
- VectorDemo.cpp
- Much better than arrays!
- Again, see CPlusPlus.com for more info and member functions.