Lecture 3: Applications of Looping and Conditional Execution

1 Monte Carlo Simulations

(from Wikipedia, retrieved on Feb. 9, 2014) Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results; typically one runs simulations many times over in order to obtain the distribution of an unknown probabilistic entity. The name comes from the resemblance of the technique to the act of playing and recording your results in a real gambling casino. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to obtain a closed-form expression, or infeasible to apply a deterministic algorithm.

1.1 Computing the Value of $\pi$

One can try to approximate the value of $\pi$ using geometry.

Assume that the side of the square above is 1 unit length.
The area of the square is then $1 \times 1 = 1$.
What is the area of the red part (a quarter of a disk)? The whole disk would have an area of $\pi r^2$, since we only have a quarter of it, the area is $\pi r^2 / 4 = \pi / 4$.
The ratio of the area of the quarter disk to the area of the whole square is $\pi / 4$.
How can this be used to approximate the value of $\pi$?
Simulation If we could randomly generate points within the whole square, we can count how many of them fall within the quarter disk (how?) and then divide that number by the total number of points. If we repeat the experiment enough times, the ratio of these two numbers should be getting closer and closer to $\pi/4$. Then $\pi$ can be approximated by multiplying the ratio by 4.

Source Code See examples Computing Pi.java and Computing Pi_GUI.java in the lecture notes.

1.2 Computing the Chances of Hitting the Bullseye in a Dart Competition

Here is a dart board. We want to figure out what the chances are of hitting the bullseye (small disk in the center) and what the chances are of hitting each of the rings.

We can employ a very similar method as in the previous examples. If we simulate random dart throws (i.e., generating random points that fall within the square board) then we can simply count how many of them fall within each region.

Note: this example is not completely realistic: it does not take into account the skill of the player and it assumes that every single dart hits the board.

Source Code See examples BullseyeChance.java and BullseyeChance_GUI.java in the lecture notes.