Game source

The implementation was done based on the instructions found here: Deconstructing the Atkins Slot Machine.

Understanding the game

In an effort to understand the game I have calculated manually some of the winning combinations displayed on the table: “Combinations for each win”.

- Five *Atkins* in a row:

  $1 \times 1 \times 1 \times 1 \times 1 = 1$

- Four *Atkins* in a row:

  $1 \times 1 \times 1 \times 1 \times (32 - 1 - 3) = 28$

  All wins have to be left aligned. If four Atkins are presented on the first four reels then the last reel has 32 possible values. From those 32 we have to exclude 1 *Atkins* symbol because the resulting combination would fall in the *five Atkins in a row case.* Also the three *Steak* symbols have to be excluded, thus the minus three, because the resulting combination would be five *Steak* in a row (pays more than four *Atkins*).

- Five *Steak* in a row:

  $[(2 + 1) \times (3 + 1) \times (2 + 1) \times (2 + 1) \times (3 + 1)] - 1 = 432$
  On all reels we merge *Steak* and *Atkins* and at the end we subtract one combination that results on five *Atkins.*

- Four *Steak* in a row:

  $[(2 + 1) \times (3 + 1) \times (2 + 1) \times (2 + 1) \times (32 - 1 - 3)] - 28 = 2996$

  In this case we merge *Steak* and *Atkins* symbols on the first four reels. Then the last can have all symbols except *Steak* and *Atkins* that both fall in the five *Steak* case. But this way we also take into account the case were the first four reels are *Atkins* which pays more than four *Steaks.* Thus, we subtract from the total result: $1 \times 1 \times 1 \times 1 \times 128 = 28$.

Code Description

The code was developed on a 64-bit version of Ubuntu Linux 10.04 and it successfully compiles and links with *g++* and *icpc* (Inter compiler), on three different Linux machines and on Mac OS X 10.7 machine. The random number generator used, while simulating the game, was the *R250* pseudo random number generator by Kirkpatrick and Stoll, which has a period of almost 2^{250}. The *Makefile* produces by default an executable named *atkins*. The results are presented below. The command *./atkins -help* displays the different game options.

Classes:

- Reel
- SlotMachine
- AtkinsMachine: inherits from Slotmachine. Creates a SlotMachine with five reels of 32 stops each and 20 different win lines.

Test answers

1) The file *src/paytable.dat* has the default format for the input file. Lines starting with # are comments. Each line is split while reading using semicolon (;) as the field separator. IMPORTANT: Always end a line with (;) to ensure, that data is correctly read. White spaces before and after each Reel Strip entry are disregarded.
2) 3) 4) 5) 6) 7) Game evaluation:
Running the command: **atkins -stat -payfile paytable.dat -Nwin 1 -print 10000000 -out statistics_1.dat** will evaluate the game by setting all reels to all possible combinations and will check wins using one win line. The *-print* option will output the combination, probability and return tables at runtime. At the end of the run these table are printed along with the *scatter (Scale)* symbol table. The output of the game will be *appended* on file: *statistics.dat.* Leaving *-print* option will output everything on the standard output.
For one win line the resulting tables at the end of the run are:

Symbol	5 in a row	4 in a row	3 in a row	2 in a row	Total
Atkins	1(0.00000003)	28(0.00000083)	513(0.00001529)	1024(0.00003052)	1566(0.00004667)
Steak	431(0.00001284)	2996(0.00008929)	32480(0.00096798)	326656(0.00973511)	362563(0.01080522)
Ham	955(0.00002846)	5157(0.00015369)	42112(0.00125504)	315392(0.00939941)	363616(0.01083660)
BuffaloWings	764(0.00002277)	5348(0.00015938)	58464(0.00174236)	430080(0.01281738)	494656(0.01474190)
Sausage	1595(0.00004753)	8613(0.00025669)	54432(0.00162220)	0(0.00000000)	64640(0.00192642)
Eggs	956(0.00002849)	6692(0.00019944)	52864(0.00157547)	0(0.00000000)	60512(0.00180340)
Butter	1995(0.00005946)	10692(0.00031865)	88704(0.00264359)	0(0.00000000)	101391(0.00302169)
Cheese	1996(0.00005949)	13860(0.00041306)	85536(0.00254917)	0(0.00000000)	101392(0.00302172)
Bacon	2976(0.00008869)	20832(0.00062084)	103168(0.00307465)	0(0.00000000)	126976(0.00378418)
Mayonnaise	2980(0.00008881)	20860(0.00062168)	128736(0.00383663)	0(0.00000000)	152576(0.00454712)
Total:	14649(0.00043657)	95078(0.00283355)	647009(0.01928237)	1073152(0.03198242)	1829888(0.05453491)

Table 1: Combinations and probabilities (in parenthesis) for each win. A total of 33554432 combinations is considered.

Further results can be obtained for different numbers of win lines by changing the -Nwin line. The folder *src/results* contains results for *-Nwin* option number of win lines. Filenames: **statistics_[Nwin].dat**.

Symbol	5 in a row	4 in a row	3 in a row	2 in a row	Total
Atkins	0.00015	0.00042	0.00076	0.00015	0.00148
Steak	0.01284	0.01786	0.03872	0.02921	0.09863
Ham	0.01423	0.02305	0.03765	0.01880	0.09373
BuffaloWings	0.00683	0.01594	0.04356	0.02563	0.09196
Sausage	0.00951	0.01925	0.03244	0.00000	0.06120
Eggs	0.00570	0.01496	0.03151	0.00000	0.05217
Butter	0.00595	0.01593	0.03965	0.00000	0.06153
Cheese	0.00595	0.02065	0.03824	0.00000	0.06484
Bacon	0.00443	0.01552	0.03075	0.00000	0.05070
Mayonnaise	0.00444	0.01554	0.03837	0.00000	0.05835
Total:	0.07003	0.15912	0.33165	0.07379	0.63460

Table 2: Return of each win

Total Scatters	Pays	Combinations	Probability	Return
5	100	486	0.000014	0.001448
4	25	20898	0.000623	0.015570
3	5	353916	0.010548	0.052738
2	0	2936772	0.087523	0.000000
1	0	11853054	0.353249	0.000000

Table 3: Scatter pays

Summary:

Prob. bonus	0.011185
Line pays	0.63460
Prob. game win	0.05453
Hitrate	18.3369

Comparison table::

# win lines	time (sec)	Line pays	Hitrate
1	11.901	0.6346	18.3369
5	21.449	0.6346	3.6673
20	56.684	0.6346	0.9168

Table 4: Comparison of running times for different number of win lines. Runs were performed on a lenovo X201 (Intel i5 processor).

Simulating the game

We can perform a simulation of the game by running the command: **./atkins -sim -payfile paytable.dat -Nwin 1 -N 100 -print 10000000 -out simulation_100_1.dat**. This command will run 100 simulation steps for one win line and will output everything on simulation_[number of steps]_[number of win lines].dat. As mentioned above the number generator used was the $R250$. Although the R250 doesn't improve the speed for the same number of simulation steps, it improves the statistics compared to the rand() % (number of reels) implementation which is slightly biased towards lower numbers. Reel::spin() function spins each reel. First, a random number of spins is chosen and then a random symbol on the reel. As we can see for a simulation close to a million

Simulation:

# steps	# win lines	time (sec)	Line pays	Hitrate	Prob. bonus
100	1	0.01	0.3200	33.33	0.000
1,000	1	0.01	0.8280	18.86	0.00800
10,000	1	0.03	0.7582	17.73	0.01100
100,000	1	0.17	0.67177	18.30	0.01083
1,000,000	1	2.45	0.64162	18.29	0.01131
10,000,000	1	23.27	0.63383	18.34	0.01111
100,000	5	0.210	0.64797	3.64	0.01160
1,000,000	5	2.050	0.63522	3.67	0.11220
10,000,000	5	29.030	0.63322	3.67	0.11140
1,000,000	10	2.650	0.63003	1.84	0.01124
1,000,000	15	4.020	0.63660	1.22	0.01096
1,000,000	20	4.630	0.63353	0.92	0.01107
10,000,000	10	34.780	0.63362	1.83	0.01121
10,000,000	15	40.180	0.635255	1.22	0.11225
10,000,000	20	46.440	0.63410	0.92	0.01118

steps, the values for the *line pays*, the *hitrate* and the *bonus probability* converge to the theoretical values.
As already mentioned the use of a different random number generator could significantly reduce the simulation time. A good candidate could be the Mersenne twister pseudo random number generator. I haven't tested it on the slot machine case, but it could be the next step, if it has not already been done.

Playing the game

Command **./atkins -game -balance 10000 -Nwin 20** will start the game on all win lines. Free spins (and free spins inside free spins) have been considered. The game displays the three lines using the symbol name, the pay and win combinations per line, as well as the free spin pays. The program prompts user whether to continue or not, until the balance is zero.