The Soul of The Sims, by Will Wright
Macintosh HD:XmotiveHarness:src/Motive.c
Tuesday, January 28, 1997 / 9:25 AM

This is the prototype for the soul of The Sims, which Will Wright wrote on January 23, 1997.

I had just started working at the Maxis Core Technology Group on "Project X" aka "Dollhouse", and Will Wright brought this code in one morning, to demonstrate his design for the motives, feedback loop and failure conditions of the simulated people. While going through old papers, I ran across this print-out that I had saved, so I scanned it and cleaned the images up, and got permission from Will to publish it.

This code is a interesting example of game design, programming and prototyping techniques. The Sims code has certainly changed a lot since Will wrote this original prototype code. For example, there is no longer any "stress" motive. And the game doesn't store motives in global variables, of course.

My hope is that this code will give you a glimpse of how Will Wright designs games, and what was going on in his head at the time!

-Don Hopkins

```
//            Motive.c        -WRW    1/23/97

#include "SRand.h"
#include  "utilities.h"

void  SimMotives(int  count);
void  ChangeMotive(int  motive,  float  value);
void  SimJob(int  type);

void  AdjustMotives(int  x,  int  y);
void  DrawMotiveSheet(void);
void  DrawMotive(int  xpos,  int  ypos,  int  value);
void  InitMotives(void);

float  Motive[16]  =  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float  oldMotive[16]  =  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};     // used for delta tests

int ClockH = 8, ClockM = 0;

enum
{
      mHappyLife      =0,
      mHappyWeek    =1,
      mHappyDay      =2,
      mHappyNow      =3,

      mPhysical=4,
      mEnergy          =5,
      mComfort=6,
      mHunger          =7,
      mHygiene =8,
      mBladder =9,

      mMental          =10,
      mAlertness      =11,
      mStress          =12,
      mEnvironment=13,
      mSocial          =14,
      mEntertained=15
};

#define DAYTICKS  720          // 1 tick = 2 minutes game time
#define WEEKTICKS  5040


void  InitMotives(void)
{
int count;

      for (count = 0; count < 16; count++)  {
            Motive[count] = 0;
      }
      Motive[mEnergy] = 70;
      Motive[mAlertness] = 20;
      Motive[mHunger] = -40;
}
```

```
void SimMotives(int count)              // simulates internal motive changes
{
float tem;
int z;
Rect r = {100,100,140,140};

    ClockM += 2;                        // inc game clock (Jamie, remove this)
    if (ClockM > 58) {
        ClockM = 0;
        ClockH++;
        if (ClockH > 24)  ClockH = 1;
    }

//  energy
    if (Motive[mEnergy] > 0)  {
        if (Motive[mAlertness] > 0)
            Motive[mEnergy]  -=  (Motive[mAlertness]/100);
        else
            Motive[mEnergy]  -=  (Motive[mAlertness]/100)  *  ((100  -  Motive[mEnergy])  /  50);
    }
    else  {
        if (Motive[mAlertness] > 0)
            Motive[mEnergy]  -=  (Motive[mAlertness]/100)  *  ((100  +  Motive[mEnergy])  /  50);
        else
            Motive[mEnergy]  -=  (Motive[mAlertness]/100);
    }

    if (Motive[mHunger] > oldMotive[mHunger])  {        // I had some food
        tem = Motive[mHunger] - oldMotive[mHunger];
        Motive[mEnergy] += tem / 4;
    }

//  comfort
    if (Motive[mBladder] < 0)
        Motive[mComfort] += Motive[mBladder] / 10; //  max  -10

    if (Motive[mHygiene] < 0)
        Motive[mComfort] += Motive[mHygiene] / 20; //  max  -5

    if (Motive[mHunger] < 0)
        Motive[mComfort] += Motive[mHunger] / 20;  //  max  -5


    // dec a max 100/cycle in a cubed curve (seek zero)
    Motive[mComfort]  -=  (Motive[mComfort]  *  Motive[mComfort]  *  Motive[mComfort])  /  10000;

//  hunger
    tem  =  ((Motive[mAlertness]+100)/200)  *  ((Motive[mHunger]+100)/100);    // ^alert * hunger^0
    Motive[mHunger] -= tem;

    if (Motive[mStress] < 0)  // stress -> hunger
        Motive[mHunger] += (Motive[mStress] / 100) * ((Motive[mHunger]+100)/100);

    if (Motive[mHunger] < -99)  {
        AlertCancel("\pYou have starved to death");
        Motive[mHunger] = 80;
```

```
    }

// hygiene
    if (Motive[mAlertness] > 0)    Motive[mHygiene] -= .3;
    else Motive[mHygiene] -= .1;

    if (Motive[mHygiene] < -97) {                          // hit limit, bath
        AlertCancel("\pYou smell very bad, mandatory bath");
        Motive[mHygiene] = 80;
    }

// bladder
    if (Motive[mAlertness] > 0)    Motive[mBladder] -= .4; // bladder fills faster while awake
    else Motive[mBladder] -= .2;

    if (Motive[mHunger] > oldMotive[mHunger]) {        // food eaten goes into bladder
        tem = Motive[mHunger] - oldMotive[mHunger];
        Motive[mBladder] -= tem / 4;
    }
    if (Motive[mBladder] < -97) {                          // hit limit, gotta go
        if (Motive[mAlertness] < 0)
            AlertCancel("\pYou have wet your bed");
        else
            AlertCancel("\pYou have soiled the carpet");
        Motive[mBladder] = 90;
    }

// alertness
    if (Motive[mAlertness] > 0)    tem = (100 - Motive[mAlertness]) / 50;    // max delta at zero
    else tem = (Motive[mAlertness] + 100) / 50;

    if (Motive[mEnergy] > 0)
        if (Motive[mAlertness] > 0)
            Motive[mAlertness] += (Motive[mEnergy] / 100) * tem;
        else
            Motive[mAlertness] += (Motive[mEnergy] / 100);
    else
        if (Motive[mAlertness] > 0)
            Motive[mAlertness] += (Motive[mEnergy] / 100);
        else
            Motive[mAlertness] += (Motive[mEnergy] / 100) * tem;


    Motive[mAlertness] += (Motive[mEntertained] / 300) * tem;

    if (Motive[mBladder] < -50)
        Motive[mAlertness] -= (Motive[mBladder] / 100) * tem;

// stress

    Motive[mStress] += Motive[mComfort] / 10;        // max -10
    Motive[mStress] += Motive[mEntertained] / 10;    // max -10
    Motive[mStress] += Motive[mEnvironment] / 15;    // max -7
    Motive[mStress] += Motive[mSocial] / 20;         // max -5

    if (Motive[mAlertness] < 0)                    // cut stress while asleep
        Motive[mStress] = Motive[mStress] / 3;
```

```
      // dec a max 100/cycle in a cubed curve (seek zero)
      Motive[mStress] -= (Motive[mStress] * Motive[mStress] * Motive[mStress]) / 10000;

      if (Motive[mStress] < 0)
            if ((SRand(30) - 100) > Motive[mStress])
                  if ((SRand(30) - 100) > Motive[mStress])  {
                        AlertCancel("\pYou have lost your temper");
                        ChangeMotive(mStress, 20);
                  }


//  environment

//  social

//  entertained
      if (Motive[mAlertness] < 0)                     // cut entertained while asleep
            Motive[mEntertained] = Motive[mEntertained] / 2;


// calc physical
      tem =  Motive[mEnergy];
      tem += Motive[mComfort];
      tem += Motive[mHunger];
      tem += Motive[mHygiene];
      tem += Motive[mBladder];
      tem = tem / 5;

      if (tem > 0)  {                           // map the linear average into squared curve
            tem = 100 - tem;
            tem = (tem * tem) / 100;
            tem = 100 - tem;
      }
      else  {
            tem = 100 + tem;
            tem = (tem * tem) / 100;
            tem = tem - 100;
      }
      Motive[mPhysical] = tem;

// calc mental
      tem += Motive[mStress];             // stress counts *2
      tem += Motive[mStress];
      tem += Motive[mEnvironment];
      tem += Motive[mSocial];
      tem += Motive[mEntertained];
      tem = tem / 5;

      if (tem > 0)  {                           // map the linear average into squared curve
            tem = 100 - tem;
            tem = (tem * tem) / 100;
            tem = 100 - tem;
      }
      else  {
            tem = 100 + tem;
            tem = (tem * tem) / 100;
```

```
            tem = tem - 100;
        }
        Motive[mMental] = tem;

// calc and average happiness
// happy = mental + physical

        Motive[mHappyNow] = (Motive[mPhysical]+Motive[mMental]) / 2;
        Motive[mHappyDay] = ((Motive[mHappyDay] * (DAYTICKS-1)) + Motive[mHappyNow]) / DAYTICKS;
        Motive[mHappyWeek] = ((Motive[mHappyWeek] * (WEEKTICKS-1)) + Motive[mHappyNow]) / WEEKTICKS;
        Motive[mHappyLife] = ((Motive[mHappyLife] * 9) + Motive[mHappyWeek]) / 10;

        for (z = 0; z < 16; z++)  {
            if (Motive[z] > 100)  Motive[z] = 100;          // check for over/under flow
            if (Motive[z] < -100)  Motive[z] = -100;
            oldMotive[z] = Motive[z];                       // save set in oldMotives (for delta tests)
        }
}

void ChangeMotive(int motive, float value)  {   // use this to change motives (checks overflow)

        Motive[motive] += value;
        if (Motive[motive] > 100) Motive[motive] = 100;
        if (Motive[motive] < -100)  Motive[motive] = -100;
}

void SimJob(int type)  {   // use this to change motives (checks overflow)

        ClockH += 9;
        if (ClockH > 24)  ClockH -= 24;

        Motive[mEnergy] = ((Motive[mEnergy] + 100) * .3) - 100;
        Motive[mHunger] = -60 + SRand(20);
        Motive[mHygiene] = -70 + SRand(30);
        Motive[mBladder] = -50 + SRand(50);
        Motive[mAlertness] = 10 + SRand(10);
        Motive[mStress] = -50 + SRand(50);
}
```