```
/*******************************************************************************
 Module
   GameService.c

 Revision
   1.0.1

 Description
   This is a template file for implementing a simple service under the
   Gen2 Events and Services Framework.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
*******************************************************************************/
/*---------------------------- Include Files ----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "GameService.h"
#include "Joystick.h"
#include "EC_Button.h"
#include "DM_Display.h"
#include "LEDDisplay.h"
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include "PALmainService.h"
#include "PIC32PortHAL.h"
#include "ES_DeferRecall.h"

/*---------------------------- Module Defines ----------------------------*/
#define NUM_ROWS   24
#define NUM_COLS   32
#define DOG_WIDTH   4
#define DOG_HEIGHT  4
#define BONE_WIDTH  4
#define BONE_HEIGHT  4
#define CAT_WIDTH 4
#define CAT_HEIGHT 4

// Timer variable
#define ONE_SEC 1000

// add a deferral queue for up to 5 pending deferrals +1 to allow for overhead
static ES_Event_t DeferralQueue[5 + 1];

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static bool checkDogBounds(int8_t mvmtX, int8_t mvmtY);
static bool checkCollision(int8_t x1, int8_t y1, int8_t w1, int8_t h1, int8_t x2,
int8_t y2,int8_t w2, int8_t h2);
static void generateBone(void);
```

```
static void genNewDogPos(void);
static void moveCat(void);
void CheckJoystick(Joystick_t current_joy);
static uint8_t MyAbs(int8_t a, int8_t b);

/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
static int8_t dogXcoord;
static int8_t dogYcoord;
static int8_t boneXcoord;
static int8_t boneYcoord;
static int8_t catXcoord;
static int8_t catYcoord;
static int16_t score;

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitGameService

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Saves away the priority, and does any
     other required initialization for this service
 Notes

 Author
     J. Edward Carryer, 01/16/12, 10:00
****************************************************************************/
bool InitGameService(uint8_t Priority)
{
  ES_Event_t ThisEvent;

  MyPriority = Priority;
  /********************************************
   in here you write your initialization code
   ********************************************/

  //initial dog coordinates
  dogXcoord = 31;
  dogYcoord = 0;
  boneXcoord = 10;
  boneYcoord = 10;
  catXcoord = 5;
  catXcoord = 19;

  // initialization of audio
  PortSetup_ConfigureDigitalOutputs(_Port_B, _Pin_5); // set pin as digital output
  CHOMP = 1; // high means off, ground triggers audio

  PortSetup_ConfigureDigitalOutputs(_Port_B, _Pin_8); // set pin as digital output
  MEOW = 1; // high means off, ground triggers audio

  InitJoystickStatus();  //setup joystick -originally in ES_IN_GAME in
```

```
  // post the initial transition event
  ThisEvent.EventType = ES_INIT;
  if (ES_PostToService(MyPriority, ThisEvent) == true)
  {
    return true;
  }
  else
  {
    return false;
  }
}

/****************************************************************************
 Function
     PostGameService

 Parameters
     EF_Event_t ThisEvent ,the event to post to the queue

 Returns
     bool false if the Enqueue operation failed, true otherwise

 Description
     Posts an event to this state machine's queue
 Notes

 Author
     J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool PostGameService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunGameService

 Parameters
   ES_Event_t : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes

 Author
   J. Edward Carryer, 01/15/12, 15:23
****************************************************************************/
ES_Event_t RunGameService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
  /********************************************
   in here you write your service code
   ********************************************/
```

```
 switch (ThisEvent.EventType)
  {
    case ES_INIT:
    {


    }
    break;

     case ES_DRAW_INIT_GAME:
     {
         //draw dog, bone, and cat
        DM_DrawDogAndBone(dogXcoord,dogYcoord,boneXcoord,boneYcoord);
        DM_DrawCat(catXcoord,catYcoord);

         //display buffer updated, send new frame event to display
        ES_Event_t NewEvent;
        NewEvent.EventType = ES_UPDATED_FRAME;
        PostLEDDisplay(NewEvent);
     }

    case ES_TIMEOUT:
    {
        if (GameMode == QueryPALmainService())//only do things if we are in game mode
        {
            if(GAME_UPDATE_TIMER == ThisEvent.EventParam){ //refresh the game

                //check collision with dog and bone
                if (true ==
checkCollision(dogXcoord,dogYcoord,DOG_WIDTH,DOG_HEIGHT,boneXcoord,boneYcoord,BONE_WIDT
H,BONE_HEIGHT)){
                    CHOMP = 0; //play chomp sound
                    ES_Timer_InitTimer(CHOMP_DELAY_TIMER, 0.2*ONE_SEC);//init timer
before sound can be played again
                    score++;//increment score
                    generateBone();//generate a new bone
                }

                //check collision with dog and cat
                if (true ==
checkCollision(dogXcoord,dogYcoord,DOG_WIDTH,DOG_HEIGHT,catXcoord,catYcoord,CAT_WIDTH,C
AT_HEIGHT)){
                    MEOW = 0; //play meow sound
                    ES_Timer_InitTimer(MEOW_DELAY_TIMER, 0.2*ONE_SEC); //init timer
before sound can be played again
                    score--; //decrement score
                    genNewDogPos(); //move dog to a new position
                }


                Joystick_t JoystickValues = Check4Joystick();
                int8_t mvmtX = JoystickValues.X;
                int8_t mvmtY = JoystickValues.Y;

                if (mvmtX != 0 || mvmtY != 0){//if any of the joystick values are not
zero
                    checkDogBounds(mvmtX,mvmtY);//if new movement puts dogs within
bounds
                    moveCat();//move cat
```

```
                    DM_DrawDogAndBone(dogXcoord,dogYcoord,boneXcoord,boneYcoord);
//draw dog+bone position
                    DM_DrawCat(catXcoord,catYcoord);//draw new cat position


                    //display buffer updated, send new frame event to display
                    ES_Event_t NewEvent;
                    NewEvent.EventType = ES_UPDATED_FRAME;
                    PostLEDDisplay(NewEvent);
                }
                CheckJoystick(JoystickValues); //check if the joystick is being used
                ES_Timer_InitTimer(GAME_UPDATE_TIMER, 75); //reset timer for refreshing
the game

            }else if(GAME_END_TIMER == ThisEvent.EventParam){ //signals end of the game
                ES_Event_t NewEvent;
                NewEvent.EventType = ES_GAME_TIMEOUT; //post game timeout event
                NewEvent.EventParam = score;
                PostPALmainService(NewEvent);
                score = 0;
            }

            //timer to prevent sounds from being played too rapidly
            else if(CHOMP_DELAY_TIMER == ThisEvent.EventParam){
                CHOMP = 1;
            }

            //timer to prevent sounds from being played too rapidly
            else if(MEOW_DELAY_TIMER == ThisEvent.EventParam){
                MEOW = 1;
            }
        }
    }
    break;

    default: {}
    break;

 }


  return ReturnEvent;
}

//set the score to zero
void zeroScore(void){
    score = 0;
}

/***************************************************************************
 private functions
 ***************************************************************************/
static bool checkDogBounds(int8_t mvmtX, int8_t mvmtY){
    bool returnVal = true;

    //check x bounds
    if (dogXcoord + mvmtX >= NUM_COLS){//Out left X bounds
        returnVal = false;
        dogXcoord = NUM_COLS - 1;//set dog to leftmost position
```

```
    }else if  (dogXcoord + mvmtX -(DOG_WIDTH-1) < 0){//out right X bounds
         returnVal = false;
         dogXcoord = DOG_WIDTH-1;//set dog to rightmost position

    }else{// in X bounds
        dogXcoord += mvmtX; //add movement to current dog position
    }

    //check y bounds
    if (dogYcoord + mvmtY +(DOG_HEIGHT-1) >= NUM_ROWS){//out top Y bounds
        returnVal = false;
        dogYcoord = NUM_ROWS - DOG_HEIGHT;//set cat to topmost position
    }else if (dogYcoord + mvmtY < 0){ //out bottom Y bounds
        returnVal = false;
        dogYcoord = 0;//set cat to bottommost position
    }else{// in y bounds
        dogYcoord += mvmtY;//addmevment to current cat position
    }

    return returnVal;
}

//check if two rectangles are overlapping
//given: bottom left (X,Y) coordinate and width/height of two object you want to check
collision
static bool checkCollision(int8_t x1, int8_t y1, int8_t w1, int8_t h1, int8_t x2,
int8_t y2,int8_t w2, int8_t h2){
    bool returnVal = true;

    if (x1 - w1 > x2 || x2 - w2 > x1){//if first object is to the left or right of the
second
        returnVal = false; //not colliding
    }else if (y1 > y2 + h2 || y2 > y1+h1){ //if first object is above or below the
second
        returnVal = false;//not colliding
    }
    //otherwise they are colliding
    return returnVal;
}

//generates a new bone on the screen
static void generateBone(void){
    static uint8_t idx = 0;

    srand(idx); //random seed

    int8_t nMax = NUM_COLS - 1; //max possible value
    int8_t nMin = BONE_WIDTH; //min possible value
    boneXcoord = ( rand()%((nMax+1)-nMin) ) + nMin; //generate new x coordinate

    nMax = NUM_ROWS - BONE_HEIGHT; //max possible value
    nMin = 0;//min possible value
    boneYcoord = ( rand()%((nMax+1)-nMin) ) + nMin;//generate new y coordinate

    idx += 5; //increment seed

}

static void genNewDogPos(void){
```

```
    static uint8_t idx = 0;

    srand(idx);//random seed

    int8_t nMax = NUM_COLS - 1;//max possible value
    int8_t nMin = DOG_WIDTH; //min possible value
    dogXcoord = ( rand()%((nMax+1)-nMin) ) + nMin; //generate new x coordinate

    nMax = NUM_ROWS - DOG_HEIGHT; //max possible value
    nMin = 0; //min possible value
    dogYcoord = ( rand()%((nMax+1)-nMin) ) + nMin;//generate new y coordinate

    idx += 5;//increment seed
}

static void moveCat(void){
    static uint8_t idx = 0;

    srand(idx);//random seed

    //generates random movement in both directions
    int8_t nMax = 2;
    int8_t nMin = -2;
    int8_t mvmtX = ( rand()%((nMax+1)-nMin) ) + nMin;
    int8_t mvmtY = ( rand()%((nMax+1)-nMin) ) + nMin;

    //used to determine if cats movment is random
    nMax = 5; //chance of randomness is 1 / (nMax-nMind+1)
    nMin = 0;
    int8_t isRandom = ( rand()%((nMax+1)-nMin) ) + nMin;


    if (isRandom <= nMax-1){ //Movement is random
        //nothing to do
    } else{ //Movement is not random - moves toward player
        int8_t towardX = dogXcoord - catXcoord; //take difference in x coords
        int8_t towardY = dogYcoord - catYcoord; //take difference in y coords

        //set x movment
        if (towardX >= 0){
            mvmtX = 1;
        }else{
            mvmtX = -1;
        }
        //set y movement
        if (towardY >= 0){
            mvmtY = 1;
        }else{
            mvmtY = -1;
        }
    }

    //check if movement moves cat out of X bounds
    if (catXcoord + mvmtX >= NUM_COLS){//Out left X bounds
        catXcoord = NUM_COLS - 1;
    }else if  (catXcoord + mvmtX -(CAT_WIDTH-1) < 0){//out right X bounds
            catXcoord = CAT_WIDTH-1;
    }else{// in X bounds
        catXcoord += mvmtX;
    }
```

```
    //check if movement moves cat out of Y bounds
    if (catYcoord + mvmtY +(CAT_HEIGHT-1) >= NUM_ROWS){//out top Y bounds
        catYcoord = NUM_ROWS - CAT_HEIGHT;
    }else if (catYcoord + mvmtY < 0){ //out bottom Y bounds
        catYcoord = 0;
    }else{// in y bounds
        catYcoord += mvmtY;
    }

    idx += 5;//increment seed
}

void CheckJoystick(Joystick_t current_joy)
{
    if (GameMode == QueryPALmainService()) //only does anything when in Gamemode
    {
        static Joystick_t past_joy = { .X = 0, .Y = 0};
        if (MyAbs(past_joy.X,current_joy.X) > 0)//if the x values have changed
        {
            ES_Timer_InitTimer(IDLE_TIMER, 30*ONE_SEC);//reset idle timer
        }
        else if (MyAbs(past_joy.Y,current_joy.Y) > 0)//if the y values have changed
        {
            ES_Timer_InitTimer(IDLE_TIMER, 30*ONE_SEC);//reset idle timer
        }
        past_joy = current_joy;
    }
}

//always returns the positve difference between two values
static uint8_t MyAbs(int8_t a, int8_t b)
{
    if (a > b)
    {
      return a - b;
    }
    else
    {
      return b - a;
    }
}

/*------------------------------ Footnotes -------------------------------*/
/*------------------------------ End of file -------------------------------*/
```