# 第十二章 项目实战

本章学习目标：

* 了解SDL2库和其相关的辅助库
* 掌握构建SDL2的开发环境
* 掌握SDL2基本绘图方法
* 掌握SDL2事件监控实现方法
* 掌握SDL2简单的游戏编程
* 掌握一般C语言程序的设计思路
* 掌握程序的调试方法

## 12.1 实践题

**一、SDL2环境的搭建**

**实验目的**

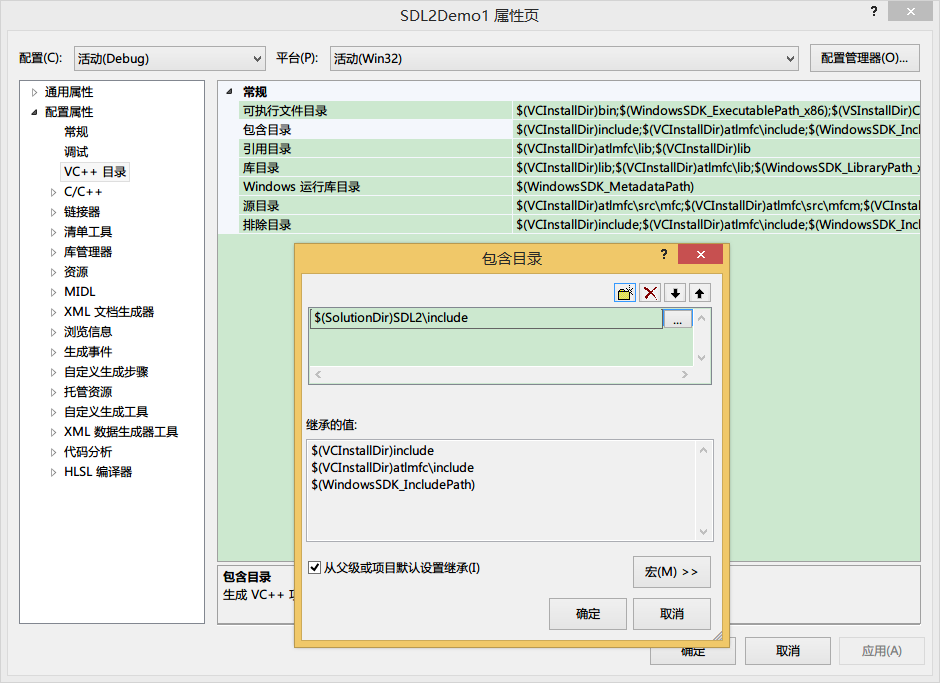
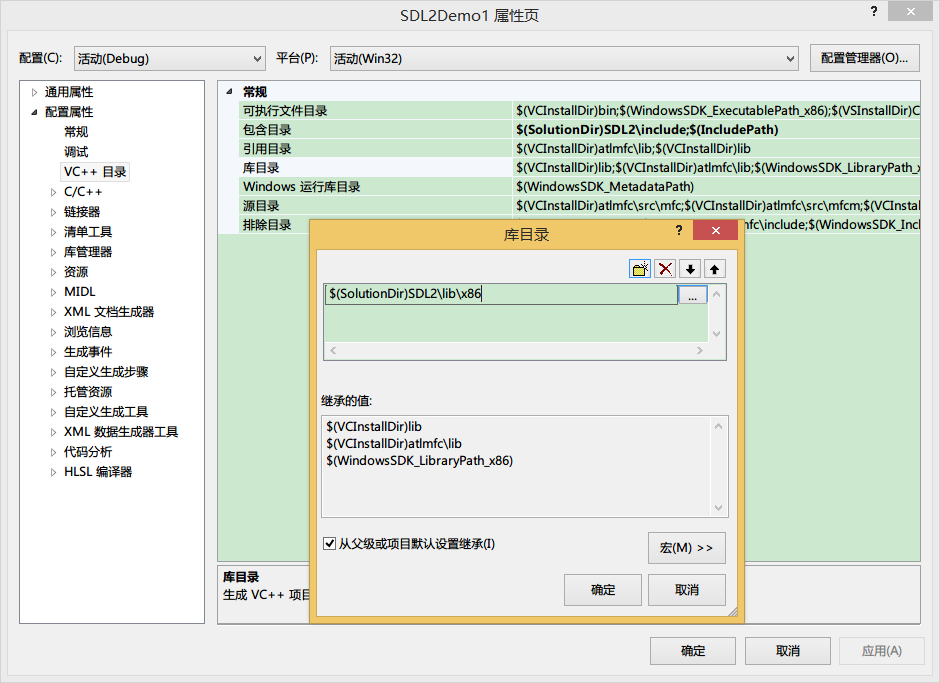
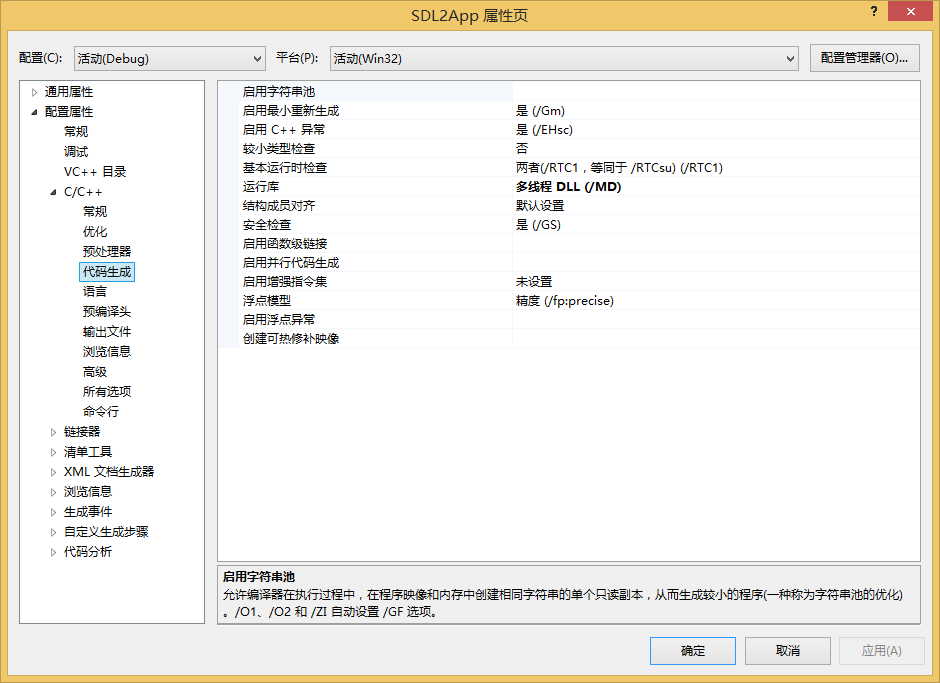
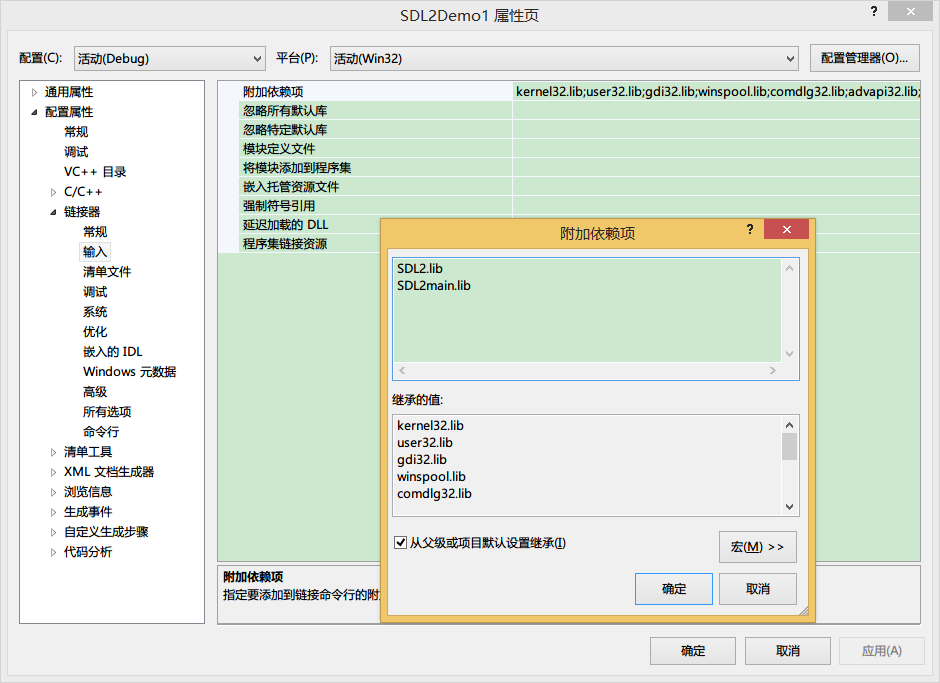
1. 掌握SDL2环境搭建
2. 测试是否搭建成功

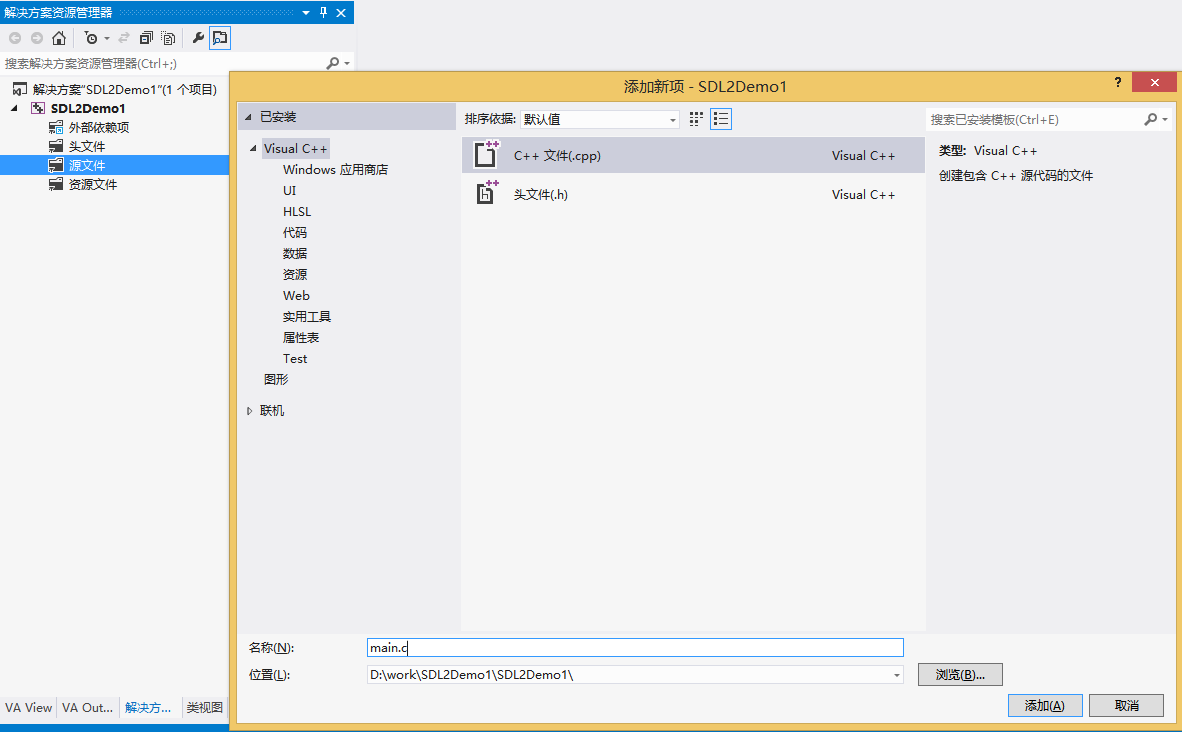
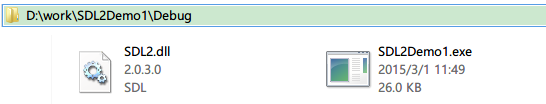
**实验步骤**

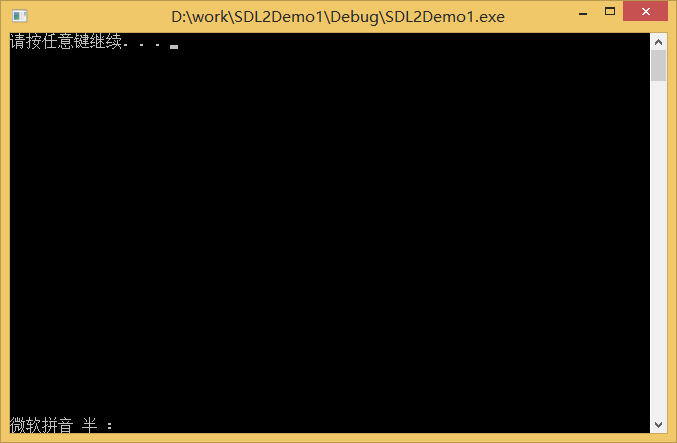
步骤1：下载SDL2-devel-2.0.3-VC.zip开发包，下载地址为<http://www.libsdl.org/release/SDL2-devel-2.0.3-VC.zip>,解压下载包。

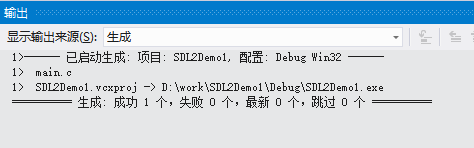
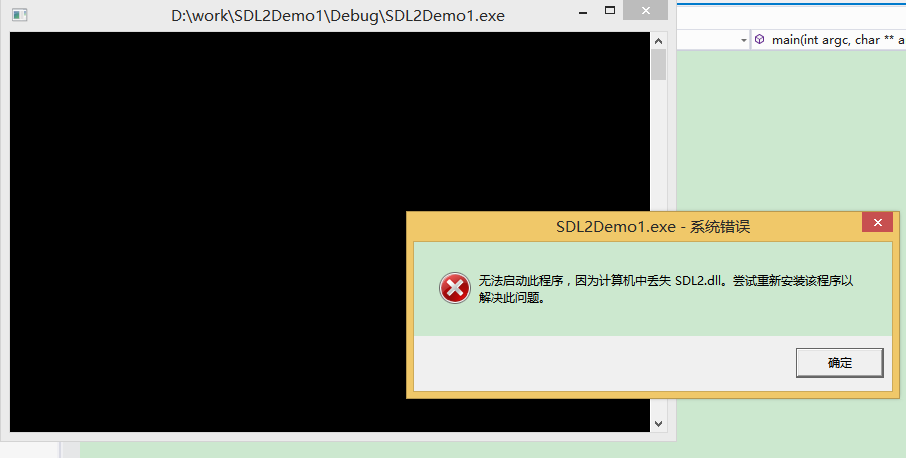
步骤2：建立解决方案和项目

打开visual studio 2012,新建win32控制台应用程序，解决方案名称为SDL2Demo1，名称为SDL2Demo1,点击确定，点击下一步，此处选择“空项目”选项，如图点击完成。

步骤3：在解决方案路径下新建SDL2文件夹，并将SDL2-devel-2.0.3-VC解压包下的include和lib文件夹拷贝到SDL2文件夹下。  
选择SDL2Demo1项目右击选择属性，导航到VC++目录，选择右侧的包含目录并新建一项，内容为$(SolutionDir)SDL2\include，点击确定。  
选择右侧的库目录并新建一项，内容为$(SolutionDir)SDL2\lib\x86，点击确定。  
  
导航到C/C++下的代码生成，右侧选择运行库改选项为”多线程 DLL (/MD)”,点击应用。  
导航到链接器下的输入选项，右侧选择附加依赖项，添加内容为SDL2.lib、SDL2main.lib，点击确定。

步骤4：在SDL2Demo1项目中选择源文件，右击添加新建项，导航到Visual C++ 下的C++文件(.cpp),名称为main.c,点击添加。  
该文件内容为：  
int main(int argc,char\*\* argv)  
{  
 return 0;  
}  
选择项目生成一下该项目，拷贝解压包SDL2-2.0.3\lib\x86路径下的SDL2.dll到解决方案的Debug目录中。

步骤5：编写测试代码，修改main.c文件中的代码  
#include <stdlib.h>  
#include <stdio.h>  
#include "SDL.h"  
  
int main(int argc,char\*\* argv)  
{  
 //初始化SDL各个子系统  
 if((SDL\_Init(SDL\_INIT\_EVERYTHING)==-1)) {  
 //初始化失败，则程序直接退出  
 exit(-1);  
}  
 //关闭SDL各个子系统   
 SDL\_Quit();   
 system("pause");   
 return 0;  
}  
编译运行程序，如果程序直接退出表示初始化失败，如果显现如下界面表示程序运行正常。

步骤6：删除解决方案目录下Debug文件夹中的SDL2.dll文件，编译项目，发现编译成功。  
运行失败。  


步骤7：选择SDL2Demo1项目的release版本进行编译运行，发现编译失败。

**实验结果/结论**

1. **实验结果**

* 实验步骤6表明程序的执行需要SDL2.dll文件。
* 实验步骤7如果要编译成功需要对release编译环境进行设置。

1. **实验结论**

* SDL2库使用的方法是导入库(lib文件)和动态库(dll文件)配合，导入库只在编译和链接阶段起作用，动态库在运行阶段起作用。
* Debug版本和Release版本都需要分别进行设置才可以编译运行。

**二、SDL2显示bmp图片**

**实验目的**

1. 掌握通过SDL2库显示一副bmp图片的流程
2. 掌握通过SDL2显示图片的一些函数

**实验步骤**

步骤1：按照试验一所示步骤搭建开发环境，建立的项目名称为SDL2App

步骤2：在解决方案目录下新建名称为res的文件夹，将准备的好的图片start.bmp和back.bmp拷贝到该目录下。

步骤3：打开main.c源文件，修改代码内容为：

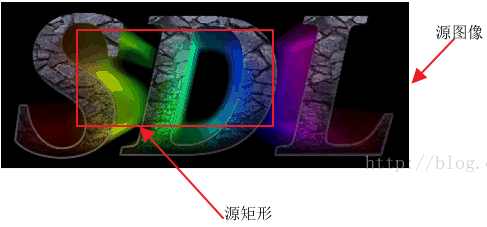
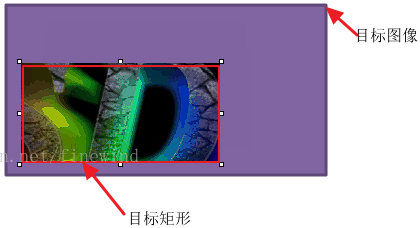
|  |
| --- |
| #include <stdlib.h>  #include <math.h>  #include <stdio.h>  #include <memory.h>  #include "SDL.h"  #define G\_WINDOW\_X 50  #define G\_WINDOW\_Y 50  #define G\_WINDOW\_W 800  #define G\_WINDOW\_H 600  typedef struct LoadedPicture {  *SDL\_Surface* \*surface;  *SDL\_Texture* \*texture;  const char\* name;  } LoadedPicture;  int *main*(int argc, char\*\* argv)  {  *Uint8* num\_pictures;  LoadedPicture\* pictures;  *SDL\_PixelFormat*\* format = *NULL*;  *SDL\_Window* \*window;  *SDL\_Renderer* \*renderer;  *Uint32* pixelFormat = 0;  int access = 0;  *SDL\_Rect* srcrect;  *SDL\_Rect* dstrect;  int i;  int j;  char \*game[] = {  "res\\back.bmp",  "res\\start.bmp"  };  *SDL\_LogSetPriority*(*SDL\_LOG\_CATEGORY\_APPLICATION*, *SDL\_LOG\_PRIORITY\_INFO*);  if (*SDL\_Init*(*SDL\_INIT\_EVERYTHING*) == -1) {  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not initialize SDL.");  *exit*(-2);  }  num\_pictures = 2;  pictures = (LoadedPicture \*)*SDL\_malloc*(sizeof(LoadedPicture)\*num\_pictures);  for (i = 0; i < num\_pictures; i++)  pictures[i].surface = *NULL*;  for (i = 0; i < num\_pictures; i++) {  pictures[i].surface = *SDL\_LoadBMP*(game[i]);  pictures[i].name = game[i];  if (pictures[i].surface == *NULL*) {  j = 0;  for (j = 0; j < num\_pictures; j++)  *SDL\_FreeSurface*(pictures[j].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not load surface from named bitmap file: %s", game[i]);  *exit*(-3);  }  }  window = *SDL\_CreateWindow*("Demo", G\_WINDOW\_X, G\_WINDOW\_Y, G\_WINDOW\_W, G\_WINDOW\_H, 0);  *SDL\_SetWindowPosition*(window, G\_WINDOW\_X, G\_WINDOW\_Y);  if (window == *NULL*) {  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create window for Demo.");  *exit*(-4);  }  renderer = *SDL\_CreateRenderer*(window, -1, 0);  if (!renderer) {  *SDL\_DestroyWindow*(window);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create rendering context for Demo window.");  *exit*(-5);  }  for (i = 0; i < num\_pictures; i++)  pictures[i].texture = *NULL*;  for (i = 0; i < num\_pictures; i++) {  pictures[i].texture = *SDL\_CreateTextureFromSurface*(renderer, pictures[i].surface);  if (pictures[i].texture == *NULL*) {  j = 0;  for (j = 0; j < num\_pictures; i++)  if (pictures[i].texture != *NULL*)  *SDL\_DestroyTexture*(pictures[i].texture);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_DestroyRenderer*(renderer);  *SDL\_DestroyWindow*(window);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create texture for Demo.");  *exit*(-6);  }  }  *SDL\_RenderCopy*(renderer, pictures[0].texture, *NULL*, *NULL*);  *SDL\_RenderPresent*(renderer);  srcrect.*x* = 10;  srcrect.*y* = 10;  srcrect.*h* = G\_WINDOW\_H / 2;  srcrect.*w* = G\_WINDOW\_W / 2;  dstrect.*x* = 10;  dstrect.*y* = 10;  dstrect.*h* = G\_WINDOW\_H / 2;  dstrect.*w* = G\_WINDOW\_W / 2;  *SDL\_RenderCopy*(renderer, pictures[1].texture, &srcrect, &dstrect);  *SDL\_RenderPresent*(renderer);  *SDL\_Delay*(2000);  srcrect.*x* = 0;  srcrect.*y* = 0;  srcrect.*h* = G\_WINDOW\_H;  srcrect.*w* = G\_WINDOW\_W;  dstrect.*x* = 0;  dstrect.*y* = 0;  dstrect.*h* = G\_WINDOW\_H;  dstrect.*w* = G\_WINDOW\_W;  *SDL\_RenderClear*(renderer);  *SDL\_RenderCopy*(renderer, pictures[0].texture, &srcrect, &dstrect);  *SDL\_RenderPresent*(renderer);  srcrect.*x* = 10;  srcrect.*y* = 10;  srcrect.*h* = G\_WINDOW\_H / 2;  srcrect.*w* = G\_WINDOW\_W / 2;  dstrect.*x* = 10;  dstrect.*y* = 10;  dstrect.*h* = G\_WINDOW\_H / 2;  dstrect.*w* = G\_WINDOW\_W / 2;  *SDL\_RenderCopy*(renderer, pictures[1].texture, &srcrect, &dstrect);  *SDL\_RenderPresent*(renderer);  *SDL\_Delay*(2000);  *system*("pause");  for (i = 0; i < num\_pictures; i++)  *SDL\_DestroyTexture*(pictures[i].texture);  *SDL\_DestroyRenderer*(renderer);  *SDL\_DestroyWindow*(window);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  return 0;  } |

**实验结果/结论**

1. **实验结果**

* 程序运行后显示界面为  
    
  延迟2秒后显示  
    
  back.bmp原始图片为  
    
  start.bmp的原始图片为  
  

1. **实验结论**

* 通过SDL2显示图片的主要流程为首先加载图片数据到SDL\_Surface然后通过SDL\_CreateTextureFromSurface创建SDL\_Texture，通过SDL\_RenderCopy和SDL\_RenderPresent显示图片，SDL使用SDL\_Surface和SDL\_Texture这2种结构绘图到屏幕。
* SDL\_RenderCopy函数，源矩形用于指定要拷贝到目标图像的源图像起始位置及宽高参数，即可以只拷贝源图像的某一部分到目标图像。  
  目标矩形用于指定要拷贝到目标图像中的起始位置及宽高参数，即可以通过指定不同的目标矩形位置，让源图像拷贝到目标图像的不同位置。且目标矩形的宽高不用和源矩形一致，这可用于缩小或放大图像。x,y为矩形的开始位置；w,h为矩形的宽高。  
    
  
* 程序启动后会显示SDL的日志窗口，便于程序开发和调试。

**三、SDL2监控键盘和鼠标事件**

**实验目的**

1. 掌握SDL2如何监控鼠标的点击事件和监控鼠标的坐标
2. 掌握SDL2如何监控键盘按键事件

**实验步骤**

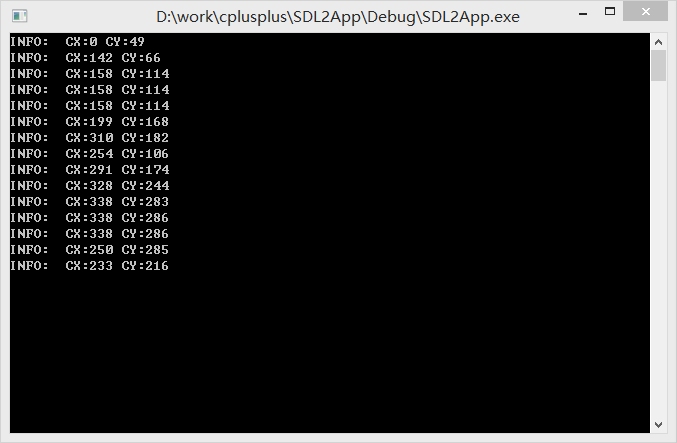
步骤1：按照试验一所示步骤搭建开发环境，建立的项目名称为SDL2CtrlApp

步骤2：打开main.c源文件，修改代码内容为：

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include "SDL.h"  #define G\_WINDOW\_X 50  #define G\_WINDOW\_Y 50  #define G\_WINDOW\_W 750  #define G\_WINDOW\_H 550  int *main*(int argc, char\*\* argv)  {  *SDL\_PixelFormat*\* format = *NULL*;  *SDL\_Window* \*window;  *SDL\_Event* event;  int event\_pending = 0;  int should\_exit = 0;  int button\_down;  int cx;  int cy;  *SDL\_LogSetPriority*(*SDL\_LOG\_CATEGORY\_APPLICATION*, *SDL\_LOG\_PRIORITY\_INFO*);  if (*SDL\_Init*(*SDL\_INIT\_EVERYTHING*) == -1) {  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not initialize SDL");  *exit*(-2);  }  window = *SDL\_CreateWindow*("Event", G\_WINDOW\_X, G\_WINDOW\_Y, G\_WINDOW\_W, G\_WINDOW\_H, 0);  *SDL\_SetWindowPosition*(window, G\_WINDOW\_X, G\_WINDOW\_Y);  event\_pending = 0;  should\_exit = 0;  event\_pending = *SDL\_PollEvent*(&event);  while (should\_exit == 0) {  event\_pending = *SDL\_PollEvent*(&event);  if (event\_pending == 1) {  if (event.*type* == *SDL\_KEYDOWN*){  button\_down = 1;  if (event.*key*.*keysym*.*sym* == *SDLK\_ESCAPE*) {  should\_exit = 1;  break;  }  }  if (event.*type* == *SDL\_MOUSEBUTTONDOWN*){  cx = event.*button*.*x*;  cy = event.*button*.*y*;  *SDL\_LogInfo*(*SDL\_LOG\_CATEGORY\_APPLICATION*, " CX:%d CY:%d ",cx, cy);  }  if (event.*type* == *SDL\_QUIT*)  should\_exit = 1;  event\_pending = 0;  }  }  *SDL\_DestroyWindow*(window);  *SDL\_Quit*();  return 0;  } |

**实验结果/结论**

1. **实验结果**

* 运行程序会启动程序的主窗口和日志窗口，在主窗口中单击鼠标会在日志窗口中提示鼠标的坐标值，这个坐标值是相对于主窗口的。  
    
  坐标通过SDL\_MouseButtonEvent结构体变量button获得。
* 将窗体控制焦点给主窗口，点击键盘的ESC键，整个程序将退出。我们通过查看SDLK\_ESCAPE的宏定义可以发现SDL\_keycode.h文件，该文件中有很多键的定义。我们程序中使用event.key.keysym.sym == SDLK\_ESCAPE进行按键的判断。

1. **实验结论**

* SDL2中事件有专门的类型SDL\_Event类型，SDL\_Event是一个联合体，其定义如下：

|  |
| --- |
| typedef union SDL\_Event  {  Uint32 type; /\*\*< Event type, shared with all events \*/  SDL\_CommonEvent common; /\*\*< Common event data \*/  SDL\_WindowEvent window; /\*\*< Window event data \*/  SDL\_KeyboardEvent key; /\*\*< Keyboard event data \*/  SDL\_TextEditingEvent edit; /\*\*< Text editing event data \*/  SDL\_TextInputEvent text; /\*\*< Text input event data \*/  SDL\_MouseMotionEvent motion; /\*\*< Mouse motion event data \*/  SDL\_MouseButtonEvent button; /\*\*< Mouse button event data \*/  SDL\_MouseWheelEvent wheel; /\*\*< Mouse wheel event data \*/  SDL\_JoyAxisEvent jaxis; /\*\*< Joystick axis event data \*/  SDL\_JoyBallEvent jball; /\*\*< Joystick ball event data \*/  SDL\_JoyHatEvent jhat; /\*\*< Joystick hat event data \*/  SDL\_JoyButtonEvent jbutton; /\*\*< Joystick button event data \*/  SDL\_JoyDeviceEvent jdevice; /\*\*< Joystick device change event data \*/  SDL\_ControllerAxisEvent caxis; /\*\*< Game Controller axis event data \*/  SDL\_ControllerButtonEvent cbutton; /\*\*< Game Controller button event data \*/  SDL\_ControllerDeviceEvent cdevice; /\*\*< Game Controller device event data \*/  SDL\_QuitEvent quit; /\*\*< Quit request event data \*/  SDL\_UserEvent user; /\*\*< Custom event data \*/  SDL\_SysWMEvent syswm; /\*\*< System dependent window event data \*/  SDL\_TouchFingerEvent tfinger; /\*\*< Touch finger event data \*/  SDL\_MultiGestureEvent mgesture; /\*\*< Gesture event data \*/  SDL\_DollarGestureEvent dgesture; /\*\*< Gesture event data \*/  SDL\_DropEvent drop; /\*\*< Drag and drop event data \*/  /\* This is necessary for ABI compatibility between Visual C++ and GCC  Visual C++ will respect the push pack pragma and use 52 bytes for  this structure, and GCC will use the alignment of the largest datatype  within the union, which is 8 bytes.  So... we'll add padding to force the size to be 56 bytes for both.  \*/  Uint8 padding[56];  } SDL\_Event; |

其中type字段决定了是那种事件，是一个枚举类型，其定义为：

|  |
| --- |
| typedef enum  {  SDL\_FIRSTEVENT = 0, /\*\*< Unused (do not remove) \*/  /\* Application events \*/  SDL\_QUIT = 0x100, /\*\*< User-requested quit \*/  /\* These application events have special meaning on iOS, see README-ios.txt for details \*/  SDL\_APP\_TERMINATING, /\*\*< The application is being terminated by the OS  Called on iOS in applicationWillTerminate()  Called on Android in onDestroy()  \*/  SDL\_APP\_LOWMEMORY, /\*\*< The application is low on memory, free memory if possible.  Called on iOS in applicationDidReceiveMemoryWarning()  Called on Android in onLowMemory()  \*/  SDL\_APP\_WILLENTERBACKGROUND, /\*\*< The application is about to enter the background  Called on iOS in applicationWillResignActive()  Called on Android in onPause()  \*/  SDL\_APP\_DIDENTERBACKGROUND, /\*\*< The application did enter the background and may not get CPU for some time  Called on iOS in applicationDidEnterBackground()  Called on Android in onPause()  \*/  SDL\_APP\_WILLENTERFOREGROUND, /\*\*< The application is about to enter the foreground  Called on iOS in applicationWillEnterForeground()  Called on Android in onResume()  \*/  SDL\_APP\_DIDENTERFOREGROUND, /\*\*< The application is now interactive  Called on iOS in applicationDidBecomeActive()  Called on Android in onResume()  \*/  /\* Window events \*/  SDL\_WINDOWEVENT = 0x200, /\*\*< Window state change \*/  SDL\_SYSWMEVENT, /\*\*< System specific event \*/  /\* Keyboard events \*/  SDL\_KEYDOWN = 0x300, /\*\*< Key pressed \*/  SDL\_KEYUP, /\*\*< Key released \*/  SDL\_TEXTEDITING, /\*\*< Keyboard text editing (composition) \*/  SDL\_TEXTINPUT, /\*\*< Keyboard text input \*/  /\* Mouse events \*/  SDL\_MOUSEMOTION = 0x400, /\*\*< Mouse moved \*/  SDL\_MOUSEBUTTONDOWN, /\*\*< Mouse button pressed \*/  SDL\_MOUSEBUTTONUP, /\*\*< Mouse button released \*/  SDL\_MOUSEWHEEL, /\*\*< Mouse wheel motion \*/  /\* Joystick events \*/  SDL\_JOYAXISMOTION = 0x600, /\*\*< Joystick axis motion \*/  SDL\_JOYBALLMOTION, /\*\*< Joystick trackball motion \*/  SDL\_JOYHATMOTION, /\*\*< Joystick hat position change \*/  SDL\_JOYBUTTONDOWN, /\*\*< Joystick button pressed \*/  SDL\_JOYBUTTONUP, /\*\*< Joystick button released \*/  SDL\_JOYDEVICEADDED, /\*\*< A new joystick has been inserted into the system \*/  SDL\_JOYDEVICEREMOVED, /\*\*< An opened joystick has been removed \*/  /\* Game controller events \*/  SDL\_CONTROLLERAXISMOTION = 0x650, /\*\*< Game controller axis motion \*/  SDL\_CONTROLLERBUTTONDOWN, /\*\*< Game controller button pressed \*/  SDL\_CONTROLLERBUTTONUP, /\*\*< Game controller button released \*/  SDL\_CONTROLLERDEVICEADDED, /\*\*< A new Game controller has been inserted into the system \*/  SDL\_CONTROLLERDEVICEREMOVED, /\*\*< An opened Game controller has been removed \*/  SDL\_CONTROLLERDEVICEREMAPPED, /\*\*< The controller mapping was updated \*/  /\* Touch events \*/  SDL\_FINGERDOWN = 0x700,  SDL\_FINGERUP,  SDL\_FINGERMOTION,  /\* Gesture events \*/  SDL\_DOLLARGESTURE = 0x800,  SDL\_DOLLARRECORD,  SDL\_MULTIGESTURE,  /\* Clipboard events \*/  SDL\_CLIPBOARDUPDATE = 0x900, /\*\*< The clipboard changed \*/  /\* Drag and drop events \*/  SDL\_DROPFILE = 0x1000, /\*\*< The system requests a file open \*/  /\* Render events \*/  SDL\_RENDER\_TARGETS\_RESET = 0x2000, /\*\*< The render targets have been reset \*/  /\*\* Events ::SDL\_USEREVENT through ::SDL\_LASTEVENT are for your use,  \* and should be allocated with SDL\_RegisterEvents()  \*/  SDL\_USEREVENT = 0x8000,  /\*\*  \* This last event is only for bounding internal arrays  \*/  SDL\_LASTEVENT = 0xFFFF  } *SDL\_EventType*; |

SDL\_KEYDOWN表示按下某键， SDL\_KEYUP, 表示松开某键，SDL\_MOUSEMOTION,表示鼠标移动，SDL\_MOUSEBUTTONDOWN, 表示鼠标键按下, SDL\_MOUSEBUTTONUP, 表示鼠标键松开。

* event.button保存的是鼠标相关的值，该结构体定义为

|  |
| --- |
| typedef struct *SDL\_MouseButtonEvent*  {  *Uint32* type; /\*\*< ::SDL\_MOUSEBUTTONDOWN or ::SDL\_MOUSEBUTTONUP \*/  *Uint32* timestamp;  *Uint32* windowID; /\*\*< The window with mouse focus, if any \*/  *Uint32* which; /\*\*< The mouse instance id, or SDL\_TOUCH\_MOUSEID \*/  *Uint8* button; /\*\*< The mouse button index \*/  *Uint8* state; /\*\*< ::SDL\_PRESSED or ::SDL\_RELEASED \*/  *Uint8* *clicks*; /\*\*< 1 for single-click, 2 for double-click, etc. \*/  *Uint8* padding1;  *Sint32* x; /\*\*< X coordinate, relative to window \*/  *Sint32* y; /\*\*< Y coordinate, relative to window \*/  } *SDL\_MouseButtonEvent*; |

其中x, y分别为鼠标的坐标值。

**四、SDL2实现打地鼠游戏**

**实验目的**

1. 掌握SDL2图像库扩展库的使用
2. 掌握使用SDL2进行小型游戏的开发

**实验步骤**

步骤1：按照试验一所示步骤搭建开发环境，建立的项目名称为WhacAMole

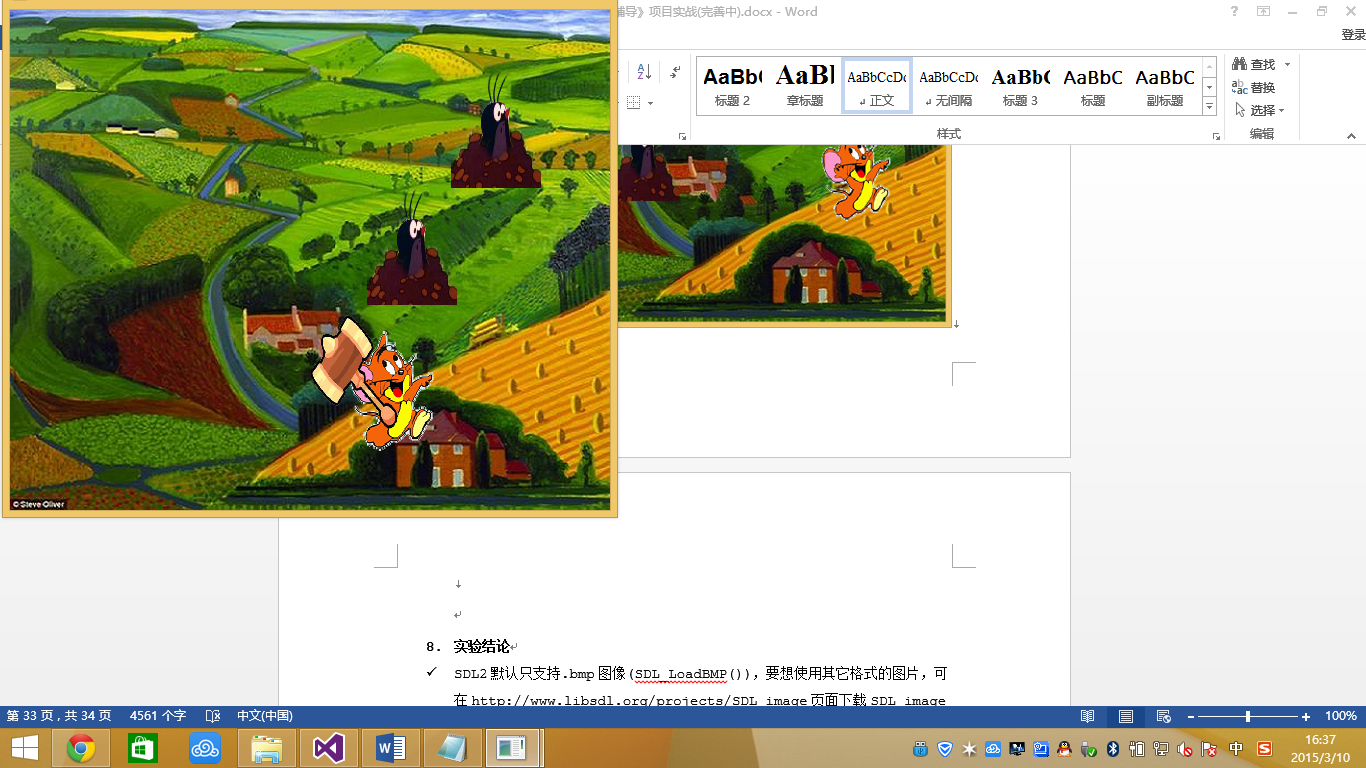
步骤2：将预先准备好的png图片素材拷贝到项目的路径下，下载SDL2的image库<http://www.libsdl.org/projects/SDL_image/release/SDL2_image-devel-2.0.0-VC.zip>并按照试验一整合到开发环境之中。

步骤3：程序的逻辑为随机的在田野中出现3只鼠，使用鼠标点击鼠，如果检测到锤子和鼠有碰撞则显示出锤子，添加main.c编写如下代码

|  |
| --- |
| #include <stdlib.h>  #include <math.h>  #include <stdio.h>  #include <memory.h>  #include <time.h>  #include "SDL.h"  #include "SDL\_image.h"  #define G\_WINDOW\_X 10  #define G\_WINDOW\_Y 10  #define G\_WINDOW\_W 600  #define G\_WINDOW\_H 500  #define G\_MOUSE\_W 90  #define G\_MOUSE\_H 120  #define G\_CUT\_W 90  #define G\_CUT\_H 120  typedef struct LoadedPicture {  *SDL\_Surface* \*surface;  *SDL\_Texture* \*texture;  const char\* name;  } LoadedPicture;  void render(*SDL\_Renderer* \*renderer, *SDL\_Texture* \*texture, *SDL\_Rect* texture\_dimensions)  {  *SDL\_RenderCopy*(renderer, texture, &texture\_dimensions, &texture\_dimensions);  *SDL\_RenderPresent*(renderer);  }  void renderTexture(*SDL\_Renderer* \*renderer, *SDL\_Texture* \*texture, int x, int y) {  *SDL\_Rect* dst;  dst.x = x;  dst.y = y;  *SDL\_QueryTexture*(texture, *NULL*, *NULL*, &dst.*w*, &dst.*h*);  *SDL\_RenderCopy*(renderer, texture, &dst, &dst);  *SDL\_RenderPresent*(renderer);  }  int *main*(int argc, char\*\* argv)  {  *Uint8* num\_pictures;  LoadedPicture\* pictures;  *SDL\_PixelFormat*\* format = *NULL*;  *SDL\_Window* \*window;  *SDL\_Renderer* \*renderer;  *SDL\_Color* black = { 0, 0, 0, 0xff };  *SDL\_Event* event;  int event\_pending = 0;  int should\_exit = 0;  unsigned int current\_picture;  int button\_down;  *Uint32* pixelFormat = 0;  int access = 0;  *SDL\_Rect* texture\_dimensions;  int mx;  int my;  int cx;  int cy;  int kn;  int mxx;  int myy;  int i;  int j;  int mxy[3][2];  *time\_t* start\_time;  *time\_t* end\_time;  double elapsed\_time;  char \*game[] = {  "back.png",  "mouse.png",  "cut.png",  "start.png",  "jerry.png"  };  mx = 0;  my = 0;  cx = 0;  cy = 0;  kn = 0;  *srand*((unsigned)*time*(*NULL*));  *SDL\_GL\_SetAttribute*(*SDL\_GL\_CONTEXT\_PROFILE\_MASK*, *SDL\_GL\_CONTEXT\_PROFILE\_ES*);  *SDL\_GL\_SetAttribute*(*SDL\_GL\_CONTEXT\_MAJOR\_VERSION*, 2);  *SDL\_GL\_SetAttribute*(*SDL\_GL\_CONTEXT\_MINOR\_VERSION*, 0);  *SDL\_LogSetPriority*(*SDL\_LOG\_CATEGORY\_APPLICATION*, *SDL\_LOG\_PRIORITY\_INFO*);  if (*SDL\_Init*(*SDL\_INIT\_EVERYTHING*) == -1) {  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not initialize SDL.");  *exit*(-2);  }  if (*IMG\_Init*(*IMG\_INIT\_PNG*) == -1)  {  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not initialize SDL IMG.");  *exit*(-2);  }  num\_pictures = 5;  pictures = (LoadedPicture \*)*SDL\_malloc*(sizeof(LoadedPicture)\*num\_pictures);  for (i = 0; i < num\_pictures; i++)  pictures[i].surface = *NULL*;  for (i = 0; i < num\_pictures; i++) {  //pictures[i].surface = SDL\_LoadBMP(game[i]);  pictures[i].surface = *IMG\_Load*(game[i]);  pictures[i].name = game[i];  if (pictures[i].surface == *NULL*) {  j = 0;  for (j = 0; j < num\_pictures; j++)  *SDL\_FreeSurface*(pictures[j].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not load surface from named file: %s", game[i]);  *exit*(-3);  }  }  window = *SDL\_CreateWindow*("Demo", G\_WINDOW\_X, G\_WINDOW\_Y, G\_WINDOW\_W, G\_WINDOW\_H, 0);  //window = SDL\_CreateWindow("Whac-A-Mole",SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED,G\_WINDOW\_W,G\_WINDOW\_H,SDL\_WINDOW\_FULLSCREEN);  //window = SDL\_CreateWindow("Whac-A-Mole", SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOW\_FULLSCREEN);  if (window == *NULL*) {  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create window for SDL\_GAME.");  *exit*(-4);  }  renderer = *SDL\_CreateRenderer*(window, -1, 0);  if (!renderer) {  *SDL\_DestroyWindow*(window);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create rendering context for SDL\_GAME window.");  *exit*(-5);  }  for (i = 0; i < num\_pictures; i++)  pictures[i].texture = *NULL*;  for (i = 0; i < num\_pictures; i++) {  pictures[i].texture = *SDL\_CreateTextureFromSurface*(renderer, pictures[i].surface);  if (pictures[i].texture == *NULL*) {  j = 0;  for (j = 0; j < num\_pictures; i++)  if (pictures[i].texture != *NULL*)  *SDL\_DestroyTexture*(pictures[i].texture);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *SDL\_DestroyRenderer*(renderer);  *SDL\_DestroyWindow*(window);  *SDL\_Quit*();  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not create texture for SDL\_shape.");  *exit*(-6);  }  }  event\_pending = 0;  should\_exit = 0;  event\_pending = *SDL\_PollEvent*(&event);  current\_picture = 0;  button\_down = 0;  texture\_dimensions.*h* = 0;  texture\_dimensions.*w* = 0;  texture\_dimensions.x = 0;  texture\_dimensions.y = 0;  *SDL\_QueryTexture*(pictures[3].texture, (*Uint32* \*)&pixelFormat, (int \*)&access, &texture\_dimensions.*w*, &texture\_dimensions.*h*);  render(renderer, pictures[3].texture, texture\_dimensions);  *SDL\_RenderClear*(renderer);  *SDL\_RenderCopy*(renderer, pictures[3].texture, &texture\_dimensions, &texture\_dimensions);  *SDL\_Delay*(2 \* 1000);  *SDL\_QueryTexture*(pictures[current\_picture].texture, (*Uint32* \*)&pixelFormat, (int \*)&access, &texture\_dimensions.*w*, &texture\_dimensions.*h*);  *SDL\_RenderClear*(renderer);  *SDL\_RenderCopy*(renderer, pictures[current\_picture].texture, &texture\_dimensions, &texture\_dimensions);  *time*(&start\_time);  while (should\_exit == 0) {  *time*(&end\_time);  elapsed\_time = *difftime*(end\_time, start\_time);  if (elapsed\_time > 2)  {  *time*(&start\_time);  texture\_dimensions.*h* = G\_WINDOW\_H;  texture\_dimensions.*w* = G\_WINDOW\_W;  texture\_dimensions.x = 0;  texture\_dimensions.y = 0;  *SDL\_RenderClear*(renderer);  *SDL\_QueryTexture*(pictures[0].texture, (*Uint32* \*)&pixelFormat, (int \*)&access, &texture\_dimensions.*w*, &texture\_dimensions.*h*);  texture\_dimensions.x = 0;  texture\_dimensions.y = 0;  *SDL\_RenderCopy*(renderer, pictures[0].texture, &texture\_dimensions, &texture\_dimensions);  for (i = 0; i < 3; i++)  {  mx = *rand*() % (G\_WINDOW\_W - G\_MOUSE\_W);  my = *rand*() % (G\_WINDOW\_H - G\_MOUSE\_H);  texture\_dimensions.*h* = G\_MOUSE\_H;  texture\_dimensions.*w* = G\_MOUSE\_W;  texture\_dimensions.x = mx;  texture\_dimensions.y = my;  if (i == 1)  {  *SDL\_RenderCopy*(renderer, pictures[4].texture, *NULL*, &texture\_dimensions);  }  else{  *SDL\_RenderCopy*(renderer, pictures[1].texture, *NULL*, &texture\_dimensions);  }  mxy[i][0] = mx;  mxy[i][1] = my;  }  *SDL\_RenderPresent*(renderer);  }  event\_pending = *SDL\_PollEvent*(&event);  if (event\_pending == 1) {  if (event.*type* == *SDL\_KEYDOWN*){  button\_down = 1;  if (event.*key*.*keysym*.*sym* == *SDLK\_ESCAPE* || event.*key*.*keysym*.*sym* == *SDLK\_KP\_BACKSPACE*) {  should\_exit = 1;  break;  }  }  if (button\_down && event.*type* == *SDL\_KEYUP*) {  *SDL\_LogInfo*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Changing to shaped bmp: %s", pictures[current\_picture].name);  }  if (event.*type* == *SDL\_MOUSEBUTTONDOWN*){  cx = event.*button*.x;  cy = event.*button*.y;  for (i = 0; i < 3; i++)  {  mx = mxy[i][0];  my = mxy[i][1];  mxx = mx + G\_MOUSE\_W / 2;  myy = my + G\_MOUSE\_H / 2;  *SDL\_LogInfo*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "MX: %d CX:%d MY: %d CY:%d abs(X):%d abs(Y):%d", mx, cx, my, cy, *abs*(mx - cx), *abs*(my - cy));  if ((*abs*(cx - mxx) < (G\_MOUSE\_W + G\_CUT\_W) / 2) && (*abs*(cy - myy) < (G\_MOUSE\_H + G\_CUT\_H) / 2))  {  *SDL\_LogInfo*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "MOUSE IS KILLED");  *SDL\_RenderClear*(renderer);  *SDL\_QueryTexture*(pictures[0].texture, (*Uint32* \*)&pixelFormat, (int \*)&access, &texture\_dimensions.*w*, &texture\_dimensions.*h*);  texture\_dimensions.x = 0;  texture\_dimensions.y = 0;  *SDL\_RenderCopy*(renderer, pictures[0].texture, &texture\_dimensions, &texture\_dimensions);  for (j = 0; j < 3; j++)  {  texture\_dimensions.*h* = G\_MOUSE\_H;  texture\_dimensions.*w* = G\_MOUSE\_W;  texture\_dimensions.x = mxy[j][0];  texture\_dimensions.y = mxy[j][1];  if (j == 1)  {  *SDL\_RenderCopy*(renderer, pictures[4].texture, *NULL*, &texture\_dimensions);  }  else{  *SDL\_RenderCopy*(renderer, pictures[1].texture, *NULL*, &texture\_dimensions);  }  }  texture\_dimensions.*h* = G\_CUT\_H;  texture\_dimensions.*w* = G\_CUT\_W;  texture\_dimensions.x = cx - G\_CUT\_W / 2;  texture\_dimensions.y = cy - G\_CUT\_H / 2;  *SDL\_RenderCopy*(renderer, pictures[2].texture, *NULL*, &texture\_dimensions);  *SDL\_RenderPresent*(renderer);  }  }  }  if (event.*type* == *SDL\_QUIT*)  should\_exit = 1;  event\_pending = 0;  }  }  for (i = 0; i < num\_pictures; i++)  *SDL\_DestroyTexture*(pictures[i].texture);  *SDL\_DestroyRenderer*(renderer);  *SDL\_DestroyWindow*(window);  for (i = 0; i < num\_pictures; i++)  *SDL\_FreeSurface*(pictures[i].surface);  *SDL\_free*(pictures);  *IMG\_Quit*();  *SDL\_Quit*();  return 0;  } |

**实验结果/结论**

1. **实验结果**

* 运行程序，程序进入全屏显示，先显示一张打地鼠的欢迎图片，然后切换为一片田野，有2只地鼠和一只老鼠随机的在田野中出现。点击任何一只鼠，则会出现锤子。  
  游戏界面  
    
  击中鼠之后的界面  
  

1. **实验结论**

* SDL2默认只支持.bmp图像(SDL\_LoadBMP())，要想使用其它格式的图片，可在http://www.libsdl.org/projects/SDL\_image页面下载SDL\_image扩展组件，它支持BMP,GIF,JPEG,LBM,PCX,PNG,PNM,TGA,TIFF,WEBP,XCF,XPM,XV格式。  
  SDL\_image扩展组件的一般使用流程为首先进行初始化

|  |
| --- |
| if (*IMG\_Init*(*IMG\_INIT\_PNG*) == -1)  {  *SDL\_LogError*(*SDL\_LOG\_CATEGORY\_APPLICATION*, "Could not initialize SDL IMG.");  *exit*(-2);  } |

然后加载图像文件IMG\_Load，使用完毕执行IMG\_Quit()。