

1. (10 points) Give the output for the following program.

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <vector>
4 const int MAX = 1000;
5
6 int main() {
7     std::vector<int> vec;
8     for (int i = 0; i < MAX; ++i) {
9         vec.push_back( rand()%100 );
10    }
11    vec.push_back(2);
12    std::cout << vec.size() << std::endl;
13    std::cout << vec.capacity() << std::endl;
14 }
```

1001

1024

-
2. (10 points) Give the output for the following program.

```
1 #include <string>
2 #include <vector>
3 #include <iostream>
4 class Pokemon {
5 public:
6     Pokemon() : name() { std::cout << "default" << std::endl; }
7     Pokemon(const std::string& n) : name(n) {
8         std::cout << "conversion" << std::endl;
9     }
10    Pokemon(const Pokemon& p) : name(p.name) {
11        std::cout << "copy" << std::endl;
12    }
13 private:
14     std::string name;
15 };
16 int main() {
17     std::vector<Pokemon> pokes;
18     pokes.push_back( std::string("Larvitar") );
19     pokes.push_back( Pokemon("Steelix") );
20 }
```

conversion

copy

conversion

copy

copy

3. (10 points) Give the output for the following program. Note the use of `reserve` on line 18.

```
1 #include <string>
2 #include <vector>
3 #include <iostream>
4 class Pokemon {
5 public:
6     Pokemon( ) : name() { std::cout << "default" << std::endl; }
7     Pokemon(const std::string& n) : name(n) {
8         std::cout << "conversion" << std::endl;
9     }
10    Pokemon(const Pokemon& p) : name(p.name) {
11        std::cout << "copy" << std::endl;
12    }
13 private:
14     std::string name;
15 };
16 int main() {
17     std::vector<Pokemon> pokes;
18     pokes.reserve(2);
19     pokes.push_back(std::string("Larvitar"));
20     pokes.push_back(Pokemon("Steelix"));
21 }
```

conversion

copy

conversion

copy

4. (10 points) Convert class Clock to a GoF singleton (the one with a static class instance variable). Be sure to “Explicitly disallow the use of compiler-generated functions you do not want.”

```
1  #include <iostream>
2
3  class Clock {
4  public:
5      static Clock* getInstance() {
6          if ( !instance ) instance = new Clock;
7          return instance;
8      }
9      int getTicks() const { return ticks; }
10     void update() { ++ticks; }
11 private:
12     int ticks;
13     static Clock* instance;
14     Clock() : ticks(0) {}
15     Clock(const Clock&);
16     Clock& operator=(const Clock&);
17 };
18
19 Clock* Clock::instance = NULL;
20 int main( ) {
21     Clock* clock = Clock::getInstance();
22     clock->update();
23     std::cout << clock->getTicks() << std::endl;
24 }
```

5. (15 points)

- (a) Give the output for the following program.
- (b) Make the program more efficient by preferring initialization to assignment?

```
1 #include <string>
2 #include <vector>
3 #include <iostream>
4 class Pokemon {
5 public:
6     Pokemon( ) : name() { std::cout << "default" << std::endl; }
7     Pokemon(const std::string& n) : name(n) {
8         std::cout << "conversion" << std::endl;
9     }
10    Pokemon(const Pokemon& p) : name(p.name) {
11        std::cout << "copy" << std::endl;
12    }
13    Pokemon& operator=(const Pokemon&) {
14        std::cout << "assignment" << std::endl;
15        return *this;
16    }
17 private:
18     std::string name;
19 };
20
21 class Pokedex {
22 public:
23     Pokedex(const std::string& s) : pokemon(s) {
24         pokemon = s;
25     }
26 private:
27     Pokemon pokemon;
28 };
29
30 int main() {
31     Pokedex p("Steelix");
32 }
```

```
*****
default
conversion
assignment
*****
```

6. (15 points) Give the output for the following program.

```
1 #include <iostream>
2
3 class A {
4 public:
5     A(const std::string& n) : name(n) {}
6     const std::string& getName() const { return name; }
7     void setName(const std::string& n) { name = n; }
8
9     virtual void display() const {
10         std::cout << name << std::endl;
11     }
12     char getIt() const { return 'A'; }
13     void onlyHere() const { std::cout << "Only in A" << std::endl; }
14 private:
15     std::string name;
16 };
17
18 class B : public A {
19 public:
20     B(const std::string& name, int x) : A(name), number(x) {}
21     virtual void display() const {
22         std::cout << A::getName() << ", " << number << std::endl;
23     }
24     char getIt() const { return 'B'; }
25     void onlyHere() const { std::cout << "Only in B" << std::endl; }
26 private:
27     int number;
28 };
29
30 int main() {
31     B b("dog", 12);
32     A* x = new B("cat", 7);
33
34     b.display();
35     x->display();
36     x->onlyHere();
37     std::cout << b.getIt() << std::endl;
38     std::cout << x->getIt() << std::endl;
39 }
```

dog, 12

cat, 7

Only in A

B

A

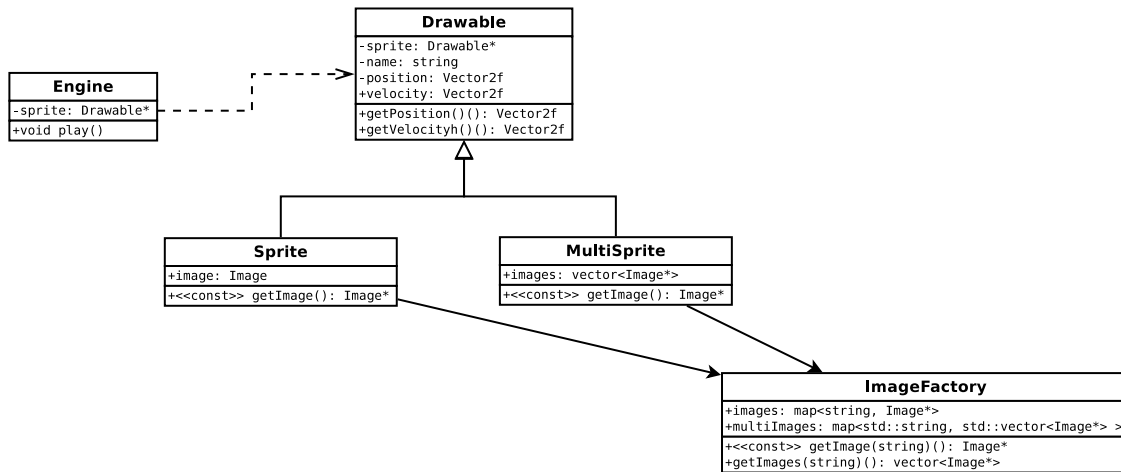


Figure 1: Partial UML class diagram for the Game Framework.

7. (10 points) Figure 1 illustrates a partial class diagram for selected classes in the game framework that we are using for project 3 and subsequent projects. Give short answers for the following questions:

(a) What two functions, listed in the class diagram, are inherited by Sprite and MultiSprite?
getPosition and getVelocity

(b) What two important *pure virtual* functions are not listed but should be included in Drawable?
draw and update

(c) Write a declaration for a *polymorphic vector* in class Engine that can contain either a Sprite or a MultiSprite.

`std::vector<Drawable*> sprites;`

(d) Notice that Sprite contains a data attribute, image, a pointer to Image. Why does a MultiSprite need a vector of pointers to Image?

A Sprite only has a single frame; however, a MultiSprite has many frames. Thus, a single instance of Image* can describe a Sprite, but a vector of Image* is needed for each frame in a multi-frame sprite.

(e) When Engine wants to create a sprite, it uses something like:

`sprite = new Sprite("star");`

How is the string "star" used so that a Sprite and an ImageFactory know the name of the file that contains the image for Sprite, the initial position and velocity of a Sprite?

The string, "star", is the name of the sprite. All of the information about a sprite is listed in an XML file and all of this information is contained in an XML record whose name is the same as the string, name, of a sprite.

8. (20 points) The program below uses SDL to draw a rectangle. Modify the program in the following way. Write a class, `Rectangle`, that contains a function `draw`, and other relevant data and operations to draw a rectangle. Then instantiate 6 rectangles of increasing size so that they appear as shown in Figure 2 (approximately). You can add a loop to main (below), and write class `Rectangle` on the following page.

```
1 #include <iostream>
2 #include <string>
3 #include <SDL2/SDL.h>
4 const int WIDTH = 640;
5 const int HEIGHT = 440;
6 const std::string TITLE = "Drawing a Rectangle";
7
8 int main (int , char*[]) {
9     if ( SDL_Init(SDL_INIT_VIDEO) != 0 ) {
10         return EXIT_FAILURE;
11     }
12     SDL_Window* window = SDL_CreateWindow(
13         TITLE.c_str(), SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
14         WIDTH, HEIGHT, SDL_WINDOW_SHOWN
15     );
16     SDL_Renderer* renderer = SDL_CreateRenderer(
17         window, -1, SDL_RENDERER_ACCELERATED
18     );
19     SDL_SetRenderDrawColor( renderer , 255, 255, 255, 255 );
20     SDL_RenderClear( renderer );
21
22     SDL_Rect r = {150, 150, 250, 150};
23     SDL_SetRenderDrawColor( renderer , 255, 0, 0, 255/2 );
24     SDL_RenderDrawRect( renderer , &r );
25     SDL_RenderPresent(renderer);
26
27     SDL_Event event;
28     const Uint8* keystate;
29     while ( true ) {
30         keystate = SDL_GetKeyboardState(0);
31         if (keystate[SDL_SCANCODE_ESCAPE]) { break; }
32         if (SDL_PollEvent(&event)) {
33             if (event.type == SDL_QUIT) {
34                 break;
35             }
36         }
37     }
38     SDL_DestroyRenderer( renderer );
39     SDL_DestroyWindow( window );
40     SDL_Quit();
41 }
```

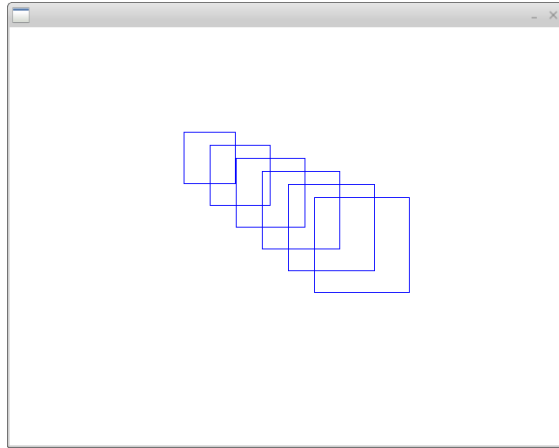


Figure 2: **Six rectangles.**

```
1 #include <iostream>
2 #include <SDL.h>
3
4 class Rectangle {
5 public:
6     Rectangle(SDL_Renderer* rend, SDL_Color c, int x, int y, int w, int h)
7         : renderer(rend), color(c), rect({x, y, w, h}) {}
8     void draw() const;
9     int getWidth() const { return rect.w; }
10    int getHeight() const { return rect.h; }
11 private:
12    SDL_Renderer* renderer;
13    SDL_Color color;
14    SDL_Rect rect;
15 };
16 std::ostream& operator<<(std::ostream&, const Rectangle&);
```

```
1 #include "rectangle.h"
2
3 void Rectangle::draw() const {
4     SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
5     SDL_RenderDrawRect(renderer, &rect);
6     SDL_RenderPresent(renderer);
7 }
8
9 std::ostream& operator<<(std::ostream& out, const Rectangle& rectangle) {
10     return out << rectangle.getWidth() << rectangle.getHeight();
11 }
```



```

1  #include <SDL2/SDL.h>
2  #include "rectangle.h"
3  #include "frameGenerator.h"
4
5  const int WINDOW_WIDTH = 640;
6  const int WINDOW_HEIGHT = 480;
7  const SDL_Color BLUE = {0,0,255,255};
8
9  int main(void) {
10     SDL_Renderer *renderer;
11     SDL_Window *window;
12
13     SDL_Init(SDL_INIT_VIDEO);
14     SDL_CreateWindowAndRenderer(
15         WINDOW_WIDTH, WINDOW_HEIGHT, 0, &window, &renderer
16     );
17
18     SDL_SetRenderDrawColor( renderer , 255, 255, 255, 255/2 );
19     SDL_RenderClear( renderer );
20
21     int w = 60, h = 60;
22     int y = 120;
23     for (int x = 200; x < 380; x+=30) {
24         SDL_Rect rect = {x, y, w, h };
25         w += 10;
26         y += 15;
27         Rectangle one(renderer , BLUE, rect.x, rect.y, rect.w, rect.h);
28         one.draw();
29     }
30
31
32     SDL_RenderPresent(renderer);
33     FrameGenerator frameGen(renderer , window, WINDOW_WIDTH, WINDOW_HEIGHT,
34         "malloy");
35     frameGen.makeFrame();
36
37     SDL_Event event;
38     const Uint8* keystate;
39     while ( true ) {
40         keystate = SDL_GetKeyboardState(0);
41         if (keystate[SDL_SCANCODE_ESCAPE]) { break; }
42         if (SDL_PollEvent(&event)) {
43             if (event.type == SDL_QUIT) {
44                 break;
45             }
46         }
47     }
48     SDL_DestroyRenderer( renderer );
49     SDL_DestroyWindow( window );
50     SDL_Quit();
51     return EXIT_SUCCESS;
52 }

```