```
#include <string>
#include <iostream>
#include <vector>
#include <sstream>
#include <SDL2/SDL.h>

using namespace std;

class AnimationFrame {
    SDL_Texture *frame;
    int time,w,h; // ms
    public:
    AnimationFrame(SDL_Texture *newFrame,int newTime=100) {
        frame=newFrame;
        time=newTime;
    }
    AnimationFrame(SDL_Renderer *ren,const char *imagePath,int newTime=100){
        SDL_Surface *bmp = SDL_LoadBMP(imagePath);
        if (bmp == NULL){
            std::cout << "SDL_LoadBMP Error: " << SDL_GetError()  << std::endl;
            SDL_Quit();
        }
        SDL_SetColorKey(bmp,SDL_TRUE,SDL_MapRGB(bmp->format,0,255,0));
        w=bmp->w;
        h=bmp->h;
        frame = SDL_CreateTextureFromSurface(ren, bmp);
        SDL_FreeSurface(bmp);
        if (frame == NULL){
            std::cout << "SDL_CreateTextureFromSurface Error: " << SDL_GetError() <<
            std::endl;
            SDL_Quit();
        }
        time=newTime;
    }
    void show(SDL_Renderer *ren,int x=0,int y=0){
      SDL_Rect src,dest;
      dest.x=x;  dest.y=y; dest.w=w; dest.h=h;
      src.x=0;  src.y=0; src.w=w; src.h=h;
      SDL_RenderCopy(ren, frame, &src, &dest);
    }
    int getTime() {
        return time;
    }
    void destroy() {
        SDL_DestroyTexture(frame);
    }
};

class Animation {
  protected:
  vector<AnimationFrame *> frames;
  int totalTime;
  public:
  Animation() {
      totalTime=0;
  }
  void addFrame(AnimationFrame *c) {
      frames.push_back(c);
      totalTime+=c->getTime();
  }
  virtual void show(SDL_Renderer *ren,int time /*ms*/,int x=0,int y=0) {
```

```
61          int aTime=time % totalTime;
62          int tTime=0;
63          unsigned int i=0;
64          for (i=0;i<frames.size();i++) {
65              tTime+=frames[i]->getTime();
66              if (aTime<=tTime) break;
67          }
68          frames[i]->show(ren,x,y);
69      }
70      virtual void destroy() {
71          for (unsigned int i=0;i<frames.size();i++)
72              frames[i]->destroy();
73      }
74  };
75
76  class Sprite : public Animation {
77      float x,dx,ax,y,dy,ay;
78      public:
79      Sprite():Animation() {
80          x=0.0; dx=0.0; ax=0.0;
81          y=0.0; dy=0.0; ay=0.0;
82      }
83      void addFrames(SDL_Renderer *ren,const char imagePath,int count) {
84      }
85
86  };
87
88  class Game {
89      protected:
90      SDL_Window *win;
91      SDL_Renderer *ren;
92      int ticks;
93      bool finished;
94      public:
95      virtual void init(const char *gameName,int maxW=640,int maxH=480,int
        startX=100,int startY=100) {
96          if (SDL_Init(SDL_INIT_VIDEO) != 0){
97              std::cout << "SDL_Init Error: " << SDL_GetError() << std::endl;
98              return;
99          }
100         win = SDL_CreateWindow(gameName, startX, startY, maxW, maxH, SDL_WINDOW_SHOWN);
101         if (win == NULL){
102             std::cout << "SDL_CreateWindow Error: " << SDL_GetError() << std::endl;
103             SDL_Quit();
104             return;
105         }
106         ren = SDL_CreateRenderer(win, -1, SDL_RENDERER_ACCELERATED |
            SDL_RENDERER_PRESENTVSYNC);
107         if (ren == NULL){
108             SDL_DestroyWindow(win);
109             std::cout << "SDL_CreateRenderer Error: " << SDL_GetError() << std::endl;
110             SDL_Quit();
111             return;
112         }
113     }
114     virtual void done() {
115         SDL_DestroyRenderer(ren);
116         SDL_DestroyWindow(win);
117         SDL_Quit();
118     }
119     void run() {
```

```
120         int start=SDL_GetTicks();
121         finished=false;
122         while(!finished) {
123           SDL_Event event;
124           if (SDL_PollEvent(&event)) {
125             if (event.type == SDL_WINDOWEVENT) {
126                 if (event.window.event==SDL_WINDOWEVENT_CLOSE)
127                   finished=true;
128             }
129             if (event.type==SDL_KEYDOWN) {
130                 if (event.key.keysym.sym==SDLK_ESCAPE)
131                   finished=true;
132             }
133             if (!finished) handleEvent(event);
134           }
135           ticks=SDL_GetTicks();
136           SDL_RenderClear(ren);
137           show();
138           SDL_RenderPresent(ren);
139         }
140         int end=SDL_GetTicks();
141         cout << "FPS "<< (300.0*1000.0/float(end-start))<<endl;
142       }
143       virtual void show()=0;
144       virtual void handleEvent(SDL_Event &event)=0;
145   };
146
147   class KarlsGame:public Game {
148       Animation a;
149       Animation world;
150       int x,y;
151       int dx,dy;
152       public:
153       void init(const char *gameName="Karl's Game",int maxW=640,int maxH=480,int
          startX=100,int startY=100){
154         Game::init(gameName);
155         a.addFrame(new AnimationFrame(ren,"hello1.bmp"));
156         a.addFrame(new AnimationFrame(ren,"hello2.bmp",500));
157         dx=1;
158         dy=1;
159         x=0;
160         y=0;
161         for (int i=1;i<=8;i++) {
162           stringstream ss;
163           ss << "Planet" << i << ".bmp";
164           world.addFrame(new AnimationFrame(ren,ss.str().c_str(),100));
165         }
166       }
167       void show() {
168           a.show(ren,ticks);
169           world.show(ren,ticks,x,y);
170           x+=dx;
171           y+=dy;
172
173       }
174       void handleEvent(SDL_Event &event) {
175       }
176       void done() {
177         a.destroy();
178         Game::done();
179       }
```

```
180  };
181
182  int main(int argc,char **argv) {
183      KarlsGame g;
184      g.init();
185      g.run();
186      g.done();
187      return 0;
188  }
189
```