

# Animation, Modules

6 - Hoạt hình, tách file

[https://github.com/csuet/AdvProg\\_AY2223](https://github.com/csuet/AdvProg_AY2223)

# Hoạt hình

- Các trò chơi trên máy tính thường không thể thiếu hoạt hình
  - <https://www.quora.com/Why-is-animation-important>
  - Trực quan, sinh động, vui
  - Dễ dàng truyền đạt thông tin, khái niệm
- Cách làm:
  - Vẽ hình
  - Đợi một lúc cho hình ảnh đọng lại trong mắt
  - Xóa màn hình và lặp lại vẽ hình kế tiếp

# Hangman 2.2 : Hoạt hình

Khi thua: hình giá treo cổ đứng đưa

Khi thắng: hình người nhảy múa

# Bắt đầu sửa từ hàm main()

```
int main()
{
    ...
    } while (badGuesses.length() < MAX_BAD_GUESSES && word !=
guessedWord);
    renderGame(guessedWord, badGuesses);
    if (badGuesses.length() < MAX_BAD_GUESSES)
        cout << "Congratulations! You win!";
    else
        cout << "You lost. The correct word is " << word;
    return 0;
}
```

# Bắt đầu sửa từ hàm main()

```
int main()  
{
```

Vùng code thông báo kết quả tại Hangman bản cũ 2.1

```
    ...  
} while (badGuesses.length() < MAX_BAD_GUESSES && word != guessedWord);  
renderGame(guessedWord, badGuesses);  
if (badGuesses.length() < MAX_BAD_GUESSES)  
    cout << "Congratulations! You win!";  
else  
    cout << "You lost. The correct word is " << word;  
    ...
```

# Bắt đầu sửa từ hàm main()

```
int main()
```

```
{
```

```
...
```

```
} while (badGuesses.length() < MAX_BAD_GUESSES && word != guessedWord);
```

```
displayFinalResult(badGuesses.length() < MAX_BAD_GUESSES, word);
```

```
...
```

```
void displayFinalResult(bool won, const string& word) {
```

```
renderGame(guessedWord, badGuesses);
```

```
if (won)
```

```
    cout << "Congratulations! You win!";
```

```
else
```

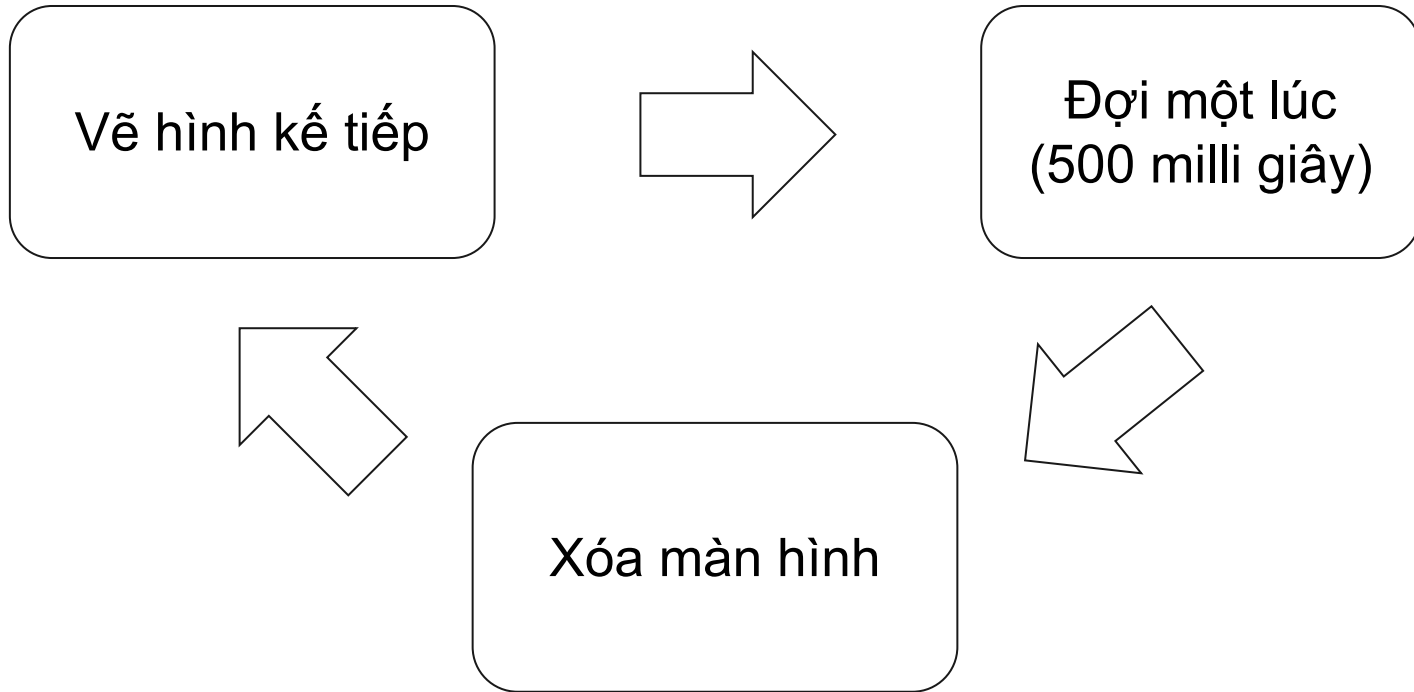
```
    cout << "You lost. The correct word is " << word;
```

```
}
```

Bắt đầu Hangman 2.2

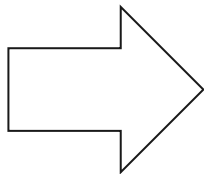
Tạm chuyển, sẽ  
thay bằng nội  
dung hoạt hình

# Cơ chế hoạt hình



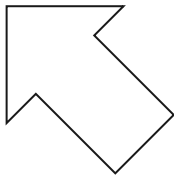
# Cơ chế hoạt hình text

```
cout << nextImage;
```

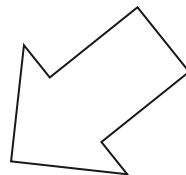


```
this_thread::sleep_for(  
    chrono::milliseconds(500));
```

```
#include <thread>  
#include <chrono>
```



```
for (int i = 0; i < 30; i++)  
    cout << endl;
```





# Thử thư viện

```
#include <iostream>
#include <thread>
#include <chrono>
using namespace std;
int main () {
```

```
    int i = 0;
```

```
    while (i < 100) {
```

```
        for (int i = 0; i < 30; i++) cout << endl;    // xóa
```

màn hình

```
        cout << i++;
```

```
        // vẽ hình kế tiếp
```

```
        this thread::sleep for(chrono::milliseconds(500));
```

Thử tạm hoạt hình các số từ 10 xuống 1  
xem thế nào

Cần dùng chuẩn **C++11**.

- Chỉnh setting CodeBlock

(Setting|Compiler...|Compiler Flags)

- Hoặc nếu biên dịch dòng lệnh cần tham số

```
C:\> g++ -std=c++11 test.cpp
```

# Phân chia mã nguồn

- Chương trình Hangman đã khá dài
  - Bắt đầu khó quản lý
  - Phần tạo animation sẽ còn dài thêm nữa.
- Phân chia mã nguồn thành nhiều mô-đun (file)
  - Dễ quản lý (mỗi mô-đun = 1 tập các hàm)
  - Có thể sử dụng lại mô-đun cho chương trình khác
  - Giảm thời gian biên dịch
    - Các tệp mã nguồn được biên dịch riêng rẽ
- Chia mô-đun theo chức năng. VD:
  - <string> chuyên xử lý chuỗi
  - <iostream> chuyên xử lý input, output

# Phân chia mã nguồn trong C++

- Mỗi mô-đun thường gồm 02 phần:
- Tập tiêu đề - header (**\*.h, \*.hpp**)
  - Khai báo hàm, khai báo kiểu, *khai báo lớp*
  - Nên viết chi tiết phạm vi để tránh nhầm lẫn
    - Ví dụ: **std::string, std::vector**
- Tập cài đặt - implementation (**\*.cpp**)
  - Cài đặt mã lệnh cho các hàm, *phương thức của lớp*
  - Có thể sử dụng lệnh **using** ở đây do biên dịch riêng
    - Ví dụ: **using namespace std; using std::string;**

# Tách code Hangman: draw.cpp

- Chuyển các *định nghĩa* hàm vẽ và dữ liệu vẽ từ hangman.cpp vào file mới **draw.cpp**
  - **void renderGame() {....}**
  - **string FIGURE[] = ....**
  - **void displayFinalResult() {...}**
- Chép các include cần thiết và khai báo namespace vào draw.cpp để giải nghĩa cho string, cout đang được dùng tại draw.cpp
  - *File nào trong chương trình C++ cũng cần có đủ các include và khai báo namespace*

# Tách code Hangman: draw.cpp

```
#include <iostream>
```

```
using namespace std;
```

```
const string FIGURE[] = {  
    "  -----  \n"  
    ...  
};
```

```
void renderGame(const string& guessedWord, const string& badGuesses)  
{  
    ...  
}
```

```
void displayFinalResult(bool won, const string& word) {  
    ...  
}
```

# Tách code Hangman: draw.h

- Chuyển các *khai báo* của các hàm vẽ từ hangman.cpp vào file mới **draw.h**
  - **void renderGame(...);**
  - **void displayFinalResult(...);**
- Chép các include cần thiết và khai báo namespace vào draw.cpp để giải nghĩa cho string, cout đang được dùng tại draw.h

```
#include <iostream>
```

```
using namespace std;
```

```
void renderGame(const string& guessedWord, const string& badGuesses);
```

```
void displayFinalResult(bool won, const string& word);
```

# Biên dịch

- Nếu biên dịch thử draw.cpp:
  - Lỗi đại loại “undefine reference to “WinMain@16” - nghĩa là không thấy hàm main, **nhưng xuất hiện file draw.o** → vậy là ổn
- Nếu biên dịch thử hangman.cpp
  - Lỗi không hiểu renderGame(), displayGameResult(). Tất nhiên, chúng được viết tại mô đun draw chứ không phải tại hangman.cpp. Trình biên dịch không ‘nhìn’ thấy.
  - Cách xử lý: nối hangman.cpp với draw
    1. Bổ sung `#include "draw.h"` tại main
    2. Dịch kèm draw.cpp, chẳng hạn bằng lệnh sau tại console:
      - a. `g++ hangman.cpp draw.cpp`

# Tạo Project

- Ta có thể tự gõ lệnh dịch phức tạp để biên dịch chương trình nhiều file. Nhưng tạo project là cách thuận tiện hơn.
- Cách làm với CodeBlocks:

File / New / Project ...

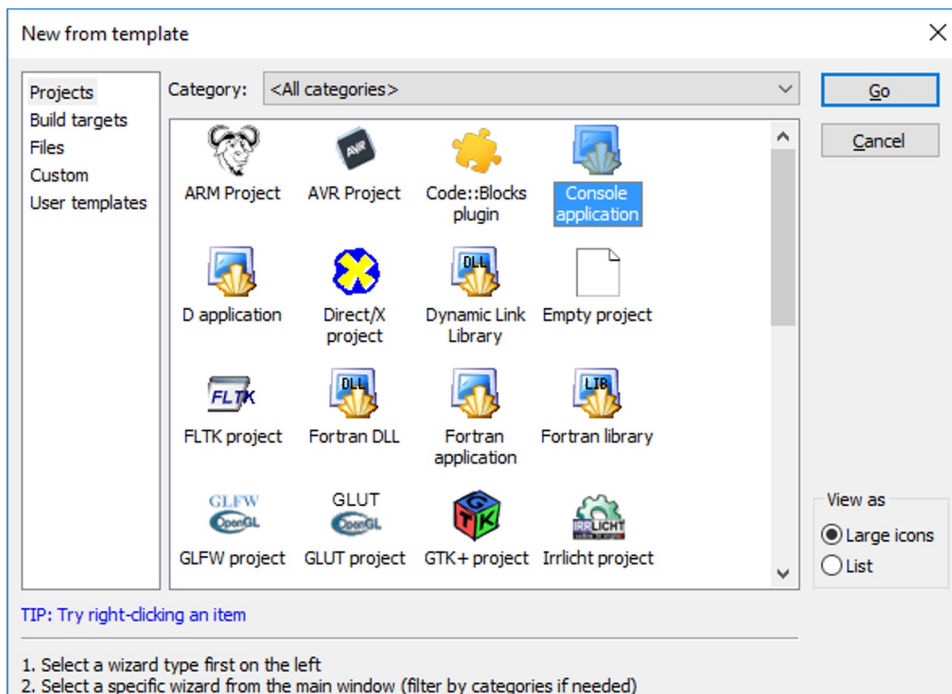
Console Application

(chương trình chạy  
trên cửa sổ lệnh)

Ấn Go

Ấn Next

Ấn Next





# Tạo Project

Chọn Project title: hangman

Chọn Thư mục chứa

**Hangman.cpp**

Ấn Next

Ấn Finish

Console application

Please select the folder where you want the new project to be created as well as its title.

Project title:  
hangman

Folder to create project in:  
D:\chauttm\1a-teaching\1.TNC\2017\sample\1ec6-anii...

Project filename:  
hangman.cbp

Resulting filename:  
D:\chauttm\1a-teaching\1.TNC\2017\sample\1ec6-animatic...

< Back   Next >   Cancel

# Tạo Project

Thêm các file **Hangman.cpp**, **draw.cpp**, **draw.h** vào Project:

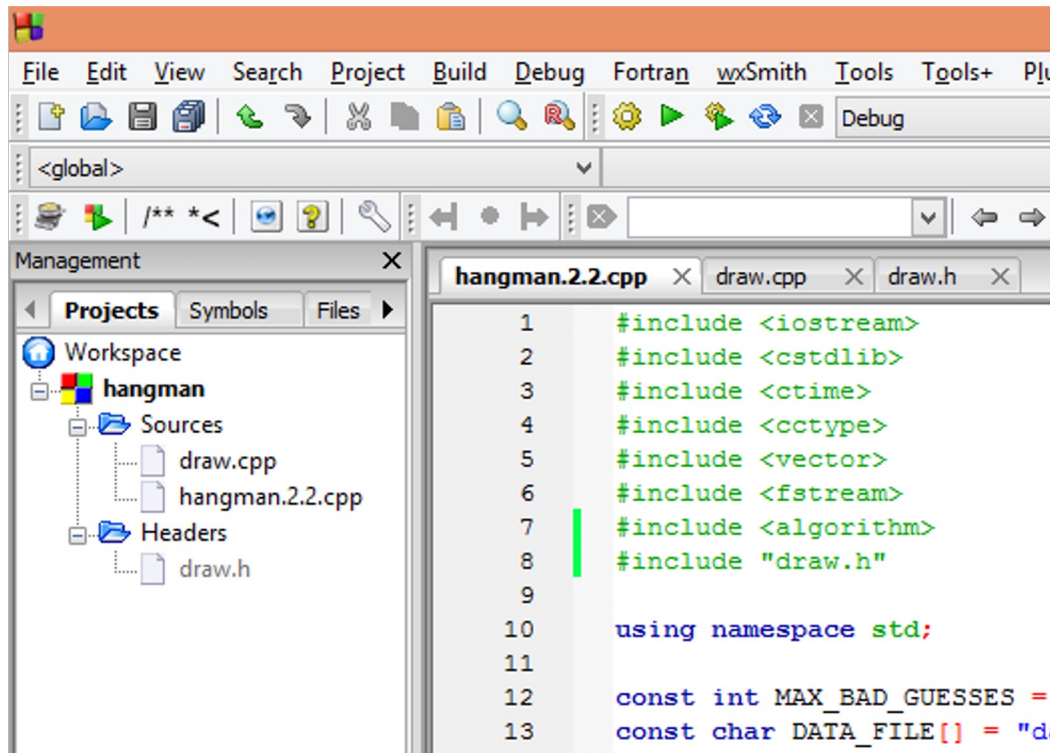
- Chọn menu

Project|Add Files..

Chọn lấy

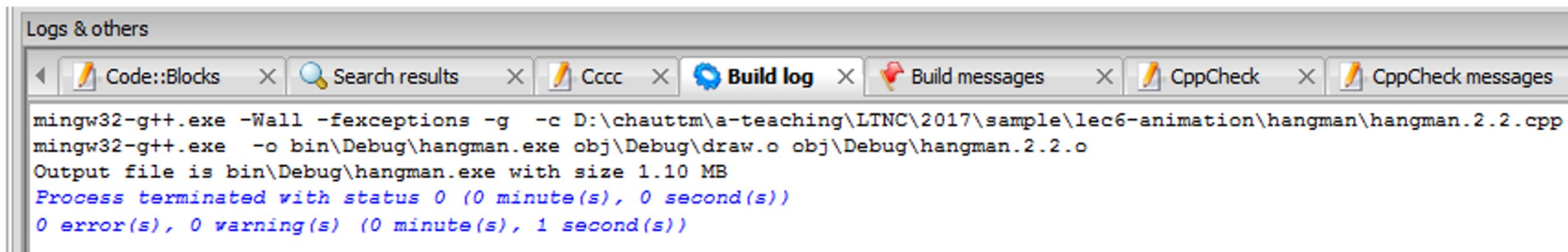
- Xóa **main.cpp** khỏi project: chuột phải vào **main.cpp** rồi Remove...

- Kết quả như hình bên



# Tạo Project

Thử dịch sẽ thấy kết quả là file **hangman.exe** tại thư mục ... **hangman\bin\Debug**



The screenshot shows the 'Logs & others' window in Visual Studio. The 'Build log' tab is selected, displaying the output of the compilation process. The log shows the use of mingw32-g++.exe with various flags to compile hangman.2.2.cpp into hangman.exe. The output file is located in bin\Debug and has a size of 1.10 MB. The process terminated successfully with status 0.

```
mingw32-g++.exe -Wall -fexceptions -g -c D:\chauttm\A-teaching\LTNC\2017\sample\lec6-animation\hangman\hangman.2.2.cpp
mingw32-g++.exe -o bin\Debug\hangman.exe obj\Debug\draw.o obj\Debug\hangman.2.2.o
Output file is bin\Debug\hangman.exe with size 1.10 MB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 1 second(s))
```

**Nhớ chạy thử xem có trục trặc gì không!**

# Đưa hoạt hình vào hangman

draw.h

```
#include <iostream>
#include <thread>
#include <chrono>
using namespace std;
int main () {
```

```
    int i = 0;
```

```
    while (i < 100) {
```

```
        for (int i = 0; i < 30; i++) cout << endl;
```

```
        cout << i++;
```

```
    this_thread::sleep_for(chrono::milliseconds(500));
```

```
}
```

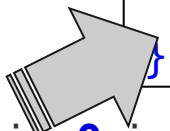
```
void displayFinalResult(bool won, const string& word) {
```

```
    if (badGuesses.length() < MAX_BAD_GUESSES)
```

```
        cout << "Congratulations! You win!";
```

```
    else
```

```
        cout << "You lost. The correct word is " << word << endl;
```



# Dura ho

draw.h

```
#include <iostream>
#include <thread>
#include <chrono>
using namespace std;
int main () {
```

```
    int i = 0;
```

```
    while (
```

```
        for
```

```
    }
}
```

```
    cout << i++;
```

```
    this_thread::sleep_for(chrono::milliseconds(500));
```

```
#include <thread>
```

```
#include <chrono>
```

```
...
```

```
void displayFinalResult(bool won, const string& word) {
```

```
    while (true) {
```

```
        for (int i = 0; i < 30; i++) cout << endl;
```

```
        if (won)
```

```
            cout << "Congratulations! You win!";
```

```
        else
```

```
            cout << "You lost. The correct word is " << word <<
```

```
            cout << (won ? getNextDancingMan() : getNextHangMan());
```

```
            this_thread::sleep_for(chrono::milliseconds(500));
```

```
    }
```

# getNextHangMan()

draw.cpp

...

```
void displayFinalResult
```

```
while (true) {
```

```
    for (int i = 0; i
```

```
    if (won)
```

```
        cout << "Congr
```

```
    else
```

```
        cout << "You lost. The correct word is << word << endl;
```

```
    cout << (won ? getNextDancingMan() : getNextHangMan());
```

```
    this_thread::sleep_for(chrono::milliseconds(500));
```

```
}
```

```
}
```

```
const string& getNextHangman()
```

```
{
```

```
    const static string figure[] = {  
        "fig1", "fig2", "fig3", "fig4"
```

```
};
```

```
const int NUMBER_OF_FIGURES =
```

```
    sizeof(figure) / sizeof(string);
```

```
static int currentFigure = 0;
```

```
return figure[(currentFigure++) % NUMBER_OF_FIGURES];
```

```
}
```

```
const string& getNextDancingman()
```

```
{
```

```
    // tương tự getNextHangman()
```

```
}
```

giữ  
currentFigure  
trong bộ nhớ

chuẩn bị  
currentFigure  
cho lần gọi sau

# Biến static

- Phạm vi nằm trong hàm
- Vòng đời dài hơn lời gọi hàm
  - Giữ nguyên giá trị giữa các lần gọi hàm.

```
void test()  
{  
    static int count = 0;  
    cout << count++;  
}  
  
int main(int argc, char* argv[])  
{  
    while (true) test();  
}
```

output

```
0  
1  
2  
3  
4  
...
```

## Biến figure của getNextHangman()

```
const static string figure[] = {  
    "   - - - - - +      \n"  
    "|         /        \n"  
    "|        O         \n"  
    "|       /|\ \      \n"  
    "|      / \ \       \n"  
    "|               \n"  
    "   - - - -      \n" ,  
    "   - - - - - +      \n"  
    "|           |       \n"  
    "|          O        \n"  
    "|         /|\ \     \n"  
    "|        / \ \      \n"  
    "|               \n"  
    "   - - - -      \n"
```

```

"    -----+      \n"
"    |          \\\n"
"    |          0      \n"
"    |        /|\\ \n"
"    |        /  \\\n"
"    |          \n"
"    -----      \n",
"    -----+      \n"
"    |          |      \n"
"    |          0      \n"
"    |        /|\\ \n"
"    |        /  \\\n"
"    |          \n"
"    -----      \n",
};

```



# Biến figure của getNextStandingman()

```
static string figure[] = {
```

```
"  0  \n"  
" /|\ \ \n"  
" | | \n",  
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",  
" _0_ \n"  
" | \n"  
" / \ \ \n",  
" \ \ 0/ \n"  
" | \n"  
" / \ \ \n",  
" _0_ \n"  
" | \n"  
" / \ \ \n",
```

```
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",  
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",  
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",  
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",  
"  0  \n"  
" /|\ \ \n"  
" / \ \ \n",
```

```
};
```

Chạy thử sẽ thấy hoạt hình đẹp hơn :-)

# Refactor

Phiên bản 2.2 với hoạt hình đơn giản đã chạy. Đến lúc dọn dẹp code → phiên bản 2.2.1

- Lặp code tại `renderGame()` và `displayGameResult()`, nên tách ra thành hàm `clearScreen()`

```
const int PATCH_LINES = 30;  
for (int i = 0; i < PATCH_LINES; i++) cout << endl;
```

- Nếu muốn gọi `clearScreen()` từ ngoài `draw.cpp`, cần bổ sung khai báo vào trong `draw.h`

```
...  
void clearScreen();  
void renderGame(const string& guessedWord, const string& badGuesses);  
void displayFinalResult(bool won, const string& word);
```

**draw.h**

# Refactor

- Code tại getNextHangMan() và getNextDancingMan() chỉ khác nhau ở biến figure[], nên gộp lại? Ví dụ

```
const string& getNextImage(const string images[], int imageCount){  
    static int currentFigure = 0;  
    return images[(currentFigure++) % imageCount];  
}
```

- Hoặc có thể bỏ hẳn và dùng kĩ thuật tại hàm renderGame()

# Bài tập

1. Tìm hiểu tiền xử lý **#ifdef ... #else ...** để phân biệt Windows và hệ điều hành khác
  - Trong Windows, còn có thể dùng **system("cls")** xóa console
  - Làm theo hướng dẫn trong <http://stackoverflow.com/questions/34842526/update-console-without-flickering-c/34843181> để xóa màn hình mà không gây nháy trong Windows
2. Sửa hàm **playAnimation()** để chạy hoạt hình trong **10 giây**