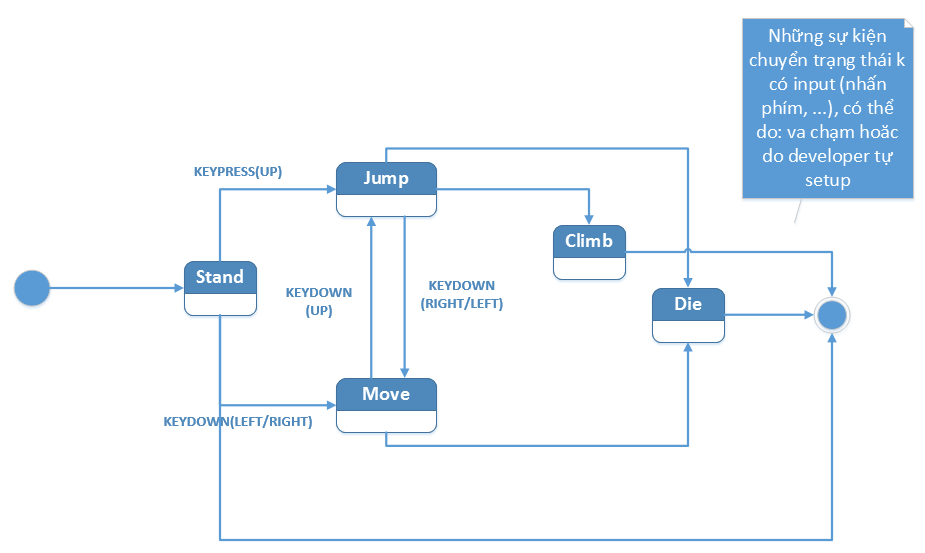
1. Bài toán

Đối với 1 đối tượng trong thực tế, có rất nhiều trạng thái (state) khác nhau. Ví dụ, con người có trạng thái: đi, đứng, ngồi, nằm,… Tương tự như thế, đối với 1 đối tượng trong game, cũng có nhiều trạng thái khác nhau. Xét về khía cạnh game, việc thay đổi trạng thái của 1 đối tượng phụ thuộc phần lớn vào thao tác input (bàn phím), hay thời gian update game (ví dụ: sau 1 khoảng thời gian nào đó, đối tượng tự động bị tiêu diệt), … Vậy làm sao để giải quyết bài toán transfer trạng thái của 1 đối tượng trong game (đặc biệt là đối với những đối tượng có những kiễu trạng thái phức tạp) vừa phụ thuộc vào input, thời gian mà vẫn đảm bảo được hiệu suất cũng như sự đơn giản hóa trong giai đoạn development và bảo trì.

Đó chỉ là vấn đề thôi, còn thì muốn tưởng tượng khúc sau thì mọi người hình dung bài toán như này (vẽ hình ra nhé): Đây là bài toán cụ thể của chúng ta, biểu diễn trạng thái của 1 con Mario:



1. Phương pháp tiếp cận

Đối với phương pháp tiếp cận Conditional statement (if – else) và Enum

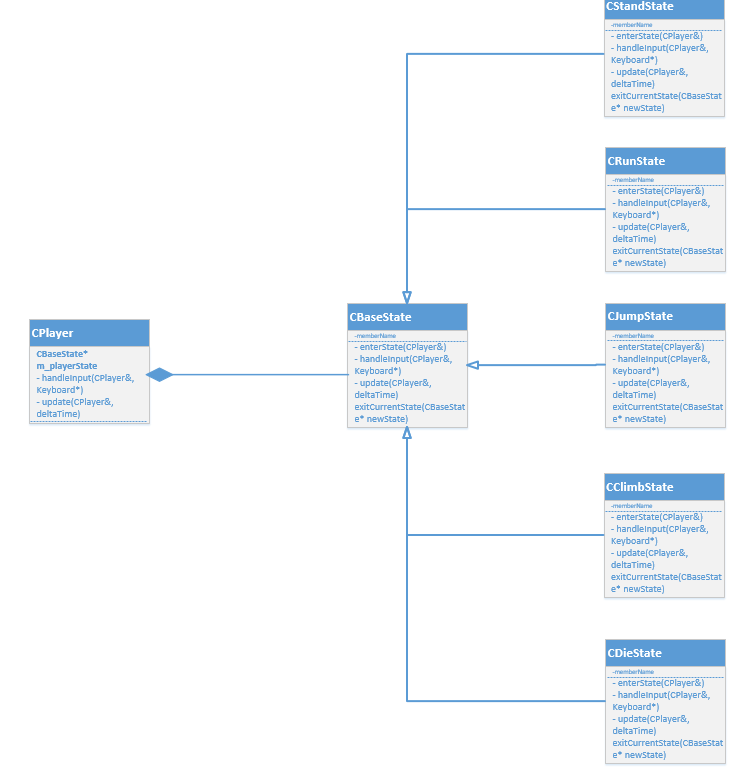
mọi người xem cái link tớ gửi khá chi tiết nhé .

<http://www.stdio.vn/articles/read/261/design-pattern-state-pattern>

Ở đây tớ trình bày cái State pattern theo project của mình

State Pattern

Đây là phương pháp thiết kế state theo tư tưởng hướng đối tượng: “*đóng gói 1 state của đối tượng thành 1 class*”, cụ thể là như sau:



* + - Tác dụng:

Thay vì tất cả mọi sự kiện liên quan đến update con player phải viết trong hàm update của đối tượng (rất phức tạp và fix bug điên đầu) thì MỖI LẦN UPDATE, PLAYER GỌI UPDATE CỦA ĐỐI TƯỢNG mBaseState. Vì là con trỏ và đa hình nên nó sẽ xác định được chính xác STATE nào của player cần được update. Cách này làm code dễ đọc hơn.

Tuy nhiên, nó có 1 hạn chế đó là: viêc đóng gói mỗi state của 1 đối tượng thành 1 class. Nếu như trong game có nhiều đối tượng update trạng thái thì phải tạo ra rất nhiều class state khác nhau. Từ đó khó quản lý.