tikzanim
Create an animation with tikz

Frédéric Bonnaud

15th of july, 2020

Contents

I apologie for my poor english.

1 Introduction

tikzanim create animations in LaTeX with tikz and animate. Steps are drawn with standard tikz code. A tikzanim's animation is made of steps. Each step draws transparencies which are uses with others to create frames of the part's step.

2 tikzanim's macros

2.1 \tikzanim

This macro uses a animateinline's environment, \multiframe and a tikzpicture's environment to prepare animationntes images).
Syntax :

```
\tikzanim <overlay> [animateinline's options] {frame rate} [tikz's options]
{initialisation} {steps}
```

- <overlay> : overlay number on which animation will be shown. Before this number, first frame of animation will be dran. After this number, last frame of animation will be drawn.

  Prefer a single number for the overlay, or animation will be render multiple times.

- [animateinline's options] : options which are pass to animateinline's environment.

  Défault value : poster=last,controls

- {frame rate} : the starting frame rate. Each step can change it.

- `[options tikz]` : options which are pass to `tikzpicture`'s environment.

  Défault value : no option

- `{initialisation}` : `tikz`'s code. This code will initialize all frames. Minimaly, it should call `\useasboundingbox` to set same size to all transparencies.

- `{étapes}` : list of steps.

2.2 `\step`

This macro define a new animation's step. `animate` uses a transparencies' stack. Each animation's frame is draw by showing all the stack. `\step` manage the stack and frame frame creation.
At start, the stack is empty. A `\step` call will :

(1) create first image of the step, put it on the stack for a frame.

(2) remove last image of the stack.

(3) create next image, put it on the stack for a frame.

(4) If it's not step's end, go to (2).

(5) If it's step's end, last image will stay on stack for some define time.

Syntaxe :
`\step[*] [frame rate] {images} [duration] {tikz' code}`

- `*` : star version will clear transparencies stack at animation's start.

- `[frame rate]` : define the new frame rate.

  Défault value : last frame rate defined either by `\tikzanim` or `\step`.

- `{images}` : define how many frames `\step` will define. If 0, a transparencies is created but no frame. The transparencies will last for `[duration]` on the stack.

- `[duration]` : set duration of last step's transparency (in frames).

  Défault value : 0, to animation's end.

- `{tikz's code}` : here goes `tikz`'s code which will draw each transparency. This code should use all previous initialised code, and, to make things depends on which frame is to be drawn, the macros `\framepos` and `\iframe`.

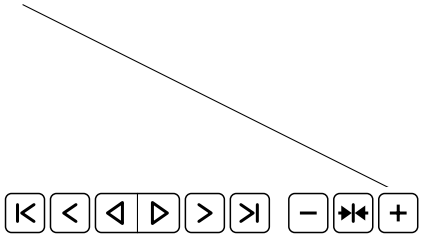
  - The macro `\framepos` is set to 0 at step's start, and will go to 1 at step's end. `\step` will set this macro.
  - The macro `\iframe` is set to transparency number. `\multiframe` from `animate`'s package will set this macro.

Le code suivant :

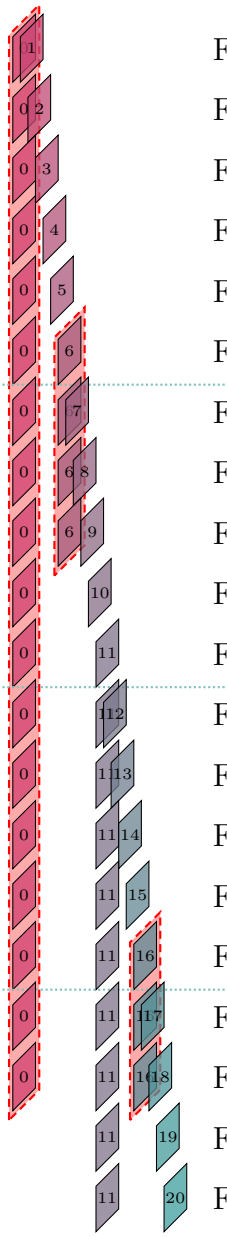
```
% \usepackage{tikzanim}
\tikzanim{10}{
  \useasboundingbox(0,0)rectangle(5,2.5) ;
}{
  % this step will last 15 frames
  % this is half next step's duration
  \step{30}[15]{
    % we need to initialize boundingbox
    \coordinate(A)at(0,0) ;
    \coordinate(B)at(5,2.5) ;
  }{
    \draw(A)--($(A)!\framepos!(B)$) ;
  }
```

```
  % next step will be 2 times speeder
  \step[20]{30}{
    % no initialisation, C and D
    % won't be use next
  }{
    \coordinate(C)at(0,2.5) ;
    \coordinate(D)at(5,0) ;
    \draw(C)--($(C)!\framepos!(D)$) ;
  }
}
```

Make this animation :



How does the stack work :



Frame 1, make of transparencies : 0,1
Frame 2, make of transparencies : 0,2
Frame 3, make of transparencies : 0,3
Frame 4, make of transparencies : 0,4
Frame 5, make of transparencies : 0,5
Frame 6, make of transparencies : 0,6

`\step{0}[18]{...}` add 1 transparency lasting 18 images, no image is added
`\step{6}[4]{...}` add 6 transparencies and 6 images, last will last 4 images

Frame 7, make of transparencies : 0,6,7
Frame 8, make of transparencies : 0,6,8
Frame 9, make of transparencies : 0,6,9
Frame 10, make of transparencies : 0,10
Frame 11, make of transparencies : 0,11

`\step{5}{...}` add 5 transparencies and 5 images

Frame 12, make of transparencies : 0,11,12
Frame 13, make of transparencies : 0,11,13
Frame 14, make of transparencies : 0,11,14
Frame 15, make of transparencies : 0,11,15
Frame 16, make of transparencies : 0,11,16

`\step{5}[3]{...}` add 5 transparencies and 5 images, last will last 3 images

Frame 17, make of transparencies : 0,11,16,17
Frame 18, make of transparencies : 0,11,16,18
Frame 19, make of transparencies : 11,19
Frame 20, make of transparencies : 11,20

`\step{4}{...}` add 4 transparencies and 4 images

2.3 `\allframes`

This macro draw on all transparencies.
Syntax :

```
\allframes {tikz's code}
```

- `{tikz's code}` : set `tikz`'s code to be drawn on each transparency. This code should use all previous initialised code, `\framepos` and `\iframe`.

3 timeline

animate ues a timeline file to manage transparencies' stack. tikzanim will create a timeline file by animation. They are named : \jobname.tzc#.tln. You can use them for debug purpose.

4 A complet sample

```
\gdef\aLen{5}
\gdef\bLen{2}
\pgfmathsetmacro{\cLen}{sqrt(\aLen^2+\bLen^2)}
\tikzset{bleu/.style={fill=blue!30,fill opacity=0.5},rouge/.style={fill=red!30,fill opacity=0.5}}
```

Some initialisations.

```

\tikzanim[poster=0,controls]{1}{
  % initialisation
  \useasboundingbox(-0.1,-\bLen-0.1)rectangle(\aLen+\bLen+0.1,\aLen+0.1) ;
}
```

Animation will start at 1 frame by seconde.
Then we create helping construction points and a transparency which will not create a frame. It only show on first frame.

```
{
  \step{0}[1]{
    \coordinate(A)at(0,0) ;
    \coordinate(B)at(\aLen,0) ;
    \coordinate(C)at(\aLen,\aLen) ;
    \coordinate(D)at(0,\aLen) ;

    \coordinate(E)at(\aLen+\bLen,0) ;
    \coordinate(F)at(\aLen+\bLen,\bLen) ;
    \coordinate(G)at(\aLen,\bLen) ;

    \path[name path=EC](E)--(C) ;
    \path[name path=FG](F)--(G) ;
    \path[name intersections={of=EC and FG,by=H}] ;

    \coordinate(I)at(0,\aLen-\bLen) ;
    \coordinate(J)at($(A)+(H)-(G)$);
    % first triangle
    \fill[bleu] (C) -- (D) -- (I) -- cycle ;
  }
```

We create a new transparency will be shown on next frame to end of the 4th step after this one.

```
  \step{0}[4.0]{
    % second triangle
    \fill[bleu] (I) -- (A) -- (J) -- cycle ;
  }
```

We create a new transparency will be shown on next frame to the end of the 5th step after this one.

```
  \step{0}[5.0]{
    % third triangle
    \fill[rouge] (E) -- (F) -- (H) -- cycle ;
  }
```

Then we create one transparency which will be shown for all next ones containing fixed stuff and the 3 first transparencies.

```
  \step{1}{
    % starting figure
```

```
  \draw (A) -- (B) -- (C) -- (D) --cycle ;
  \draw (B) -- (E) -- (F) -- (G) --cycle ;

  \fill[bleu] (I) -- (J) -- (B) -- (C) -- cycle ;
  \draw (J) -- (B) -- (C) ; \draw[densely dotted] (C) -- (I) -- (J) ;

  \fill[rouge] (B) -- (E) -- (H) -- (G) -- cycle ;
  \draw (H) -- (G) -- (B) -- (E) ; \draw[densely dotted] (E) -- (H) ;
}
```

First moving step at a frame rate of 5 images by second, for 10 transparencies, last will be lasting 1 image.

```
% first triangle moves :
\step[5]{10}[1]{
  \coordinate(M)at($(C)!\framepos!(E)$);% Point at \framepos on [CE]
  \coordinate(CM)at($(M)-(C)$) ;
  \coordinate(ICM)at($(I)+(CM)$);
  \coordinate(CCM)at($(C)+(CM)$);
  \coordinate(DCM)at($(D)+(CM)$);
  \coordinate(S)at(barycentric cs:I=1,C=1,D=1);
  \coordinate(S')at(barycentric cs:ICM=1,CCM=1,DCM=1);
  \fill[bleu] (ICM) -- (CCM) -- (DCM) -- cycle ;
  \draw[densely dotted] (ICM) -- (CCM) ;
  \draw (CCM) -- (DCM) -- (ICM) ;
  \draw[-latex](S)--(S') ;
}
```

We draw only triangle on last position (not the arrow).

```
% final postion
\step{1}{
  \fill[bleu] (ICM) -- (CCM) -- (DCM) -- cycle ;
  \draw[densely dotted] (ICM) -- (CCM) ;
  \draw (CCM) -- (DCM) -- (ICM) ;
}
```

We do the same for the second triangle.

```
% second triangle moves
\step{10}[1]{
  \coordinate(N)at($(I)!\framepos!(C)$);% Point at \framepos on [IC]
  \coordinate(IN)at($(N)-(I)$) ;
  \coordinate(IIN)at($(I)+(IN)$);
  \coordinate(JIN)at($(J)+(IN)$);
  \coordinate(AIN)at($(A)+(IN)$);
  \coordinate(T')at(barycentric cs:IIN=1,JIN=1,AIN=1);
  \fill[bleu] (IIN) -- (JIN) -- (AIN) -- cycle ;
  \draw[densely dotted] (IIN) -- (JIN) ;
  \draw (JIN) -- (AIN) -- (IIN) ;
  \draw[-latex](T)--(T') ;
}
% final position
\step{1}{
  \fill[bleu] (IIN) -- (JIN) -- (AIN) -- cycle ;
  \draw[densely dotted] (IIN) -- (JIN) ;
  \draw (JIN) -- (AIN) -- (IIN) ;
}
```

We do the same for the third triangle.

```
% third tiangle moves
\step{10}[1]{
  \coordinate(P)at($(H)!\framepos!(J)$);% Point at \framepos on [HJ]
  \coordinate(HP)at($(P)-(H)$) ;
  \coordinate(EHP)at($(E)+(HP)$);
  \coordinate(FHP)at($(F)+(HP)$);
```

```
    \coordinate(HHP)at($(H)+(HP)$);
    \coordinate(U)at(barycentric cs:E=1,F=1,H=1);
    \coordinate(U')at(barycentric cs:EHP=1,FHP=1,HHP=1);
    \fill[rouge] (EHP) -- (FHP) -- (HHP) -- cycle ;
    \draw[densely dotted] (EHP) -- (HHP) ;
    \draw (EHP) -- (FHP) -- (HHP) ;
    \draw[-latex](U)--(U') ;
  }
  % final position
  \step{1}{
    \fill[rouge] (EHP) -- (FHP) -- (HHP) -- cycle ;
    \draw[densely dotted] (EHP) -- (HHP) ;
    \draw (EHP) -- (FHP) -- (HHP) ;
  }
}
```

The animation :
A visual proof of Pythagore's theorem.