

tikzanim

Create an animation with tikz

Frédéric Bonnaud

15th of july, 2020

Contents

1	Introduction	1
2	tikzanim's macros	1
2.1	<code>\tikzanim</code>	1
2.2	<code>\step</code>	2
2.3	<code>\allframes</code>	3
3	timeline	4
4	A complet sample	4

I apologise 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 used 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 drawn. After this number, last frame of animation will be drawn.

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

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

Default 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.

Défaut value : no option

- {initialisation} : tikz's code. This code will initialize all frames. Minimally, 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.
Défaut 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).
Défaut 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 :

```
1 % \usepackage{tikzanim}
2 \tikzanim{10}{
3   \useasboundingbox(0,0)rectangle(5,2.5) ;
4 }{
5   % this step will last 15 frames
6   % this is half next step's duration
```

```
7   \step{30}[15]{
8     % we need to initialize boundingbox
9     \coordinate(A)at(0,0) ;
10    \coordinate(B)at(5,2.5) ;
11  }{
12    \draw(A)--($(A)!\framepos!(B)$) ;
13  }
```

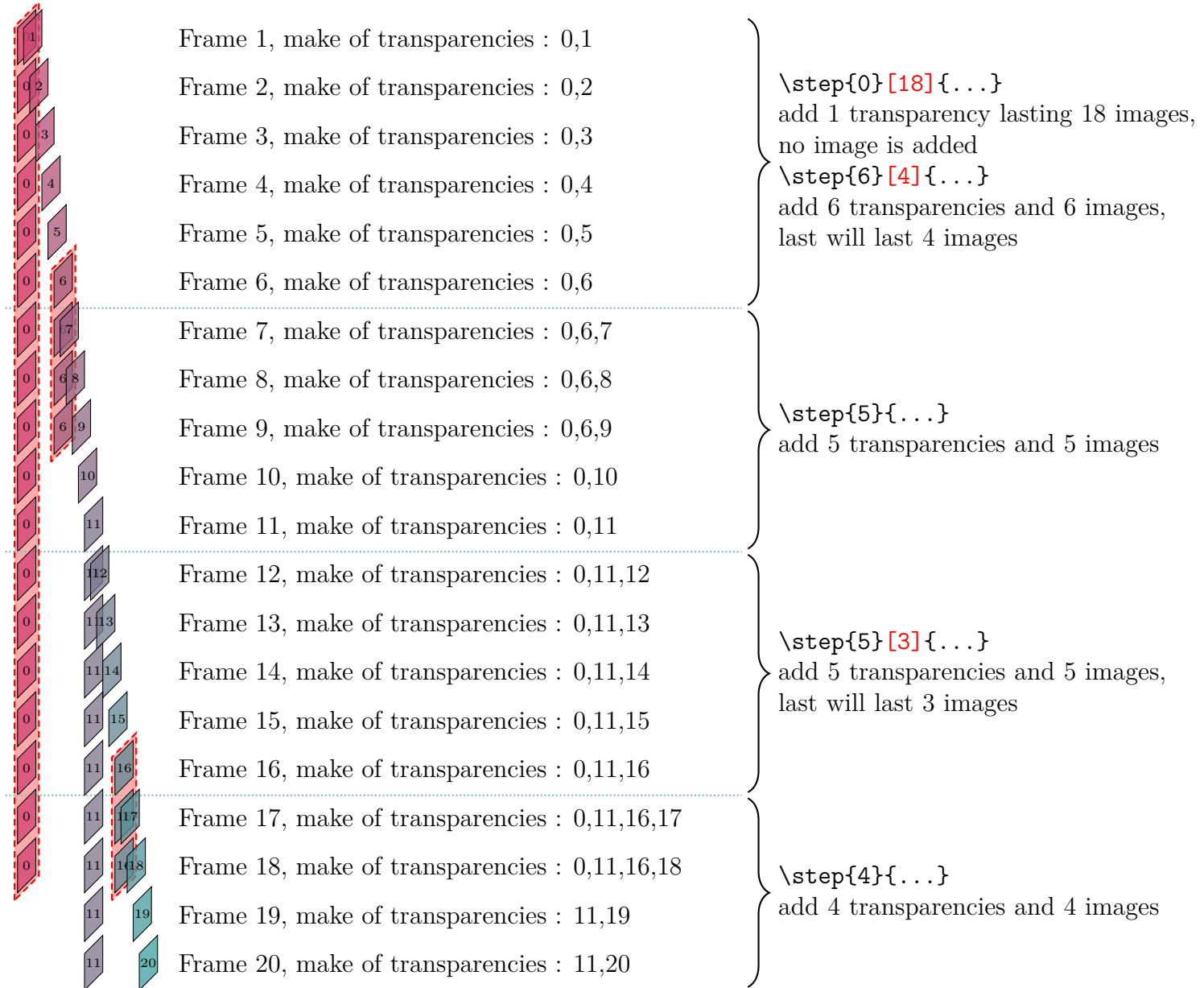
```

14 % next step will be 2 times speeder
15 \step[20]{30}{
16   % no initialisation, C and D
17   % won't be use next
18 }{
19   \coordinate(C)at(0,2.5) ;
20   \coordinate(D)at(5,0) ;
21   \draw(C)--($(C)!\framepos!(D)$) ;
22 }
23 }

```

Make this animation :

How does the stack work :



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 chargé de dessiner sur tous les transparents. This code should use all previous initialised code, `\framepos` and `\iframe`.

3 timeline

`animate` uses 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

```
1 \gdef\alen{5}
2 \gdef\blen{2}
3 \pgfmathsetmacro{\clen}{sqrt(\alen^2+\blen^2)}
4 \tikzset{bleu/.style={fill=blue!30,fill opacity=0.5},rouge/.style={fill=red!30,fill opacity=0.5}}
```

Some initialisations.

```
5
6 \tikzanim[poster=0,controls]{1}{
7   % initialisation
8   \useasboundingbox(-0.1,-\blen-0.1)rectangle(\alen+\blen+0.1,\alen+0.1) ;
9 }
```

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.

```
10 {
11   \step{0}[1]{
12     \coordinate(A)at(0,0) ;
13     \coordinate(B)at(\alen,0) ;
14     \coordinate(C)at(\alen,\alen) ;
15     \coordinate(D)at(0,\alen) ;
16
17     \coordinate(E)at(\alen+\blen,0) ;
18     \coordinate(F)at(\alen+\blen,\blen) ;
19     \coordinate(G)at(\alen,\blen) ;
20
21     \path[name path=EC](E)--(C) ;
22     \path[name path=FG](F)--(G) ;
23     \path[name intersections={of=EC and FG,by=H}] ;
24
25     \coordinate(I)at(0,\alen-\blen) ;
26     \coordinate(J)at($(A)+(H)-(G)$) ;
27     % first triangle
28     \fill[bleu] (C) -- (D) -- (I) -- cycle ;
29   }
```

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

```
30   \step{0}[4.0]{
31     % second triangle
32     \fill[bleu] (I) -- (A) -- (J) -- cycle ;
33   }
```

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

```
34   \step{0}[5.0]{
35     % third triangle
36     \fill[rouge] (E) -- (F) -- (H) -- cycle ;
37   }
```

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

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

```

40 \draw (A) -- (B) -- (C) -- (D) --cycle ;
41 \draw (B) -- (E) -- (F) -- (G) --cycle ;
42
43 \fill[bleu] (I) -- (J) -- (B) -- (C) -- cycle ;
44 \draw (J) -- (B) -- (C) ; \draw[densely dotted] (C) -- (I) -- (J) ;
45
46 \fill[rouge] (B) -- (E) -- (H) -- (G) -- cycle ;
47 \draw (H) -- (G) -- (B) -- (E) ; \draw[densely dotted] (E) -- (H) ;
48 }

```

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

```

49 % first triangle moves :
50 \step[5]{10}[1]{
51   \coordinate(M)at($(C)!\framepos!(E)$);% Point at \framepos on [CE]
52   \coordinate(CM)at($(M)-(C)$) ;
53   \coordinate(ICM)at($(I)+(CM)$);
54   \coordinate(CCM)at($(C)+(CM)$);
55   \coordinate(DCM)at($(D)+(CM)$);
56   \coordinate(S)at(barycentric cs:I=1,C=1,D=1);
57   \coordinate(S')at(barycentric cs:ICM=1,CCM=1,DCM=1);
58   \fill[bleu] (ICM) -- (CCM) -- (DCM) -- cycle ;
59   \draw[densely dotted] (ICM) -- (CCM) ;
60   \draw (CCM) -- (DCM) -- (ICM) ;
61   \draw[-latex] (S)--(S') ;
62 }

```

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

```

63 % final position
64 \step{1}{
65   \fill[bleu] (ICM) -- (CCM) -- (DCM) -- cycle ;
66   \draw[densely dotted] (ICM) -- (CCM) ;
67   \draw (CCM) -- (DCM) -- (ICM) ;
68 }

```

We do the same for the second triangle.

```

69 % second triangle moves
70 \step{10}[1]{
71   \coordinate(N)at($(I)!\framepos!(C)$);% Point at \framepos on [IC]
72   \coordinate(IN)at($(N)-(I)$) ;
73   \coordinate(IIN)at($(I)+(IN)$);
74   \coordinate(JIN)at($(J)+(IN)$);
75   \coordinate(AIN)at($(A)+(IN)$);
76   \coordinate(T')at(barycentric cs:IIN=1,JIN=1,AIN=1);
77   \fill[bleu] (IIN) -- (JIN) -- (AIN) -- cycle ;
78   \draw[densely dotted] (IIN) -- (JIN) ;
79   \draw (JIN) -- (AIN) -- (IIN) ;
80   \draw[-latex] (T)--(T') ;
81 }
82 % final position
83 \step{1}{
84   \fill[bleu] (IIN) -- (JIN) -- (AIN) -- cycle ;
85   \draw[densely dotted] (IIN) -- (JIN) ;
86   \draw (JIN) -- (AIN) -- (IIN) ;
87 }

```

We do the same for the third triangle.

```

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

```

```

7   \coordinate(HHP)at($(H)+(HP)$);
8   \coordinate(U)at(barycentric cs:E=1,F=1,H=1);
9   \coordinate(U')at(barycentric cs:EHP=1,FHP=1,HHP=1);
10  \fill[rouge] (EHP) -- (FHP) -- (HHP) -- cycle ;
11  \draw[densely dotted] (EHP) -- (HHP) ;
12  \draw (EHP) -- (FHP) -- (HHP) ;
13  \draw[-latex] (U)--(U') ;
14  }
15  % final position
16  \step{1}{
17    \fill[rouge] (EHP) -- (FHP) -- (HHP) -- cycle ;
18    \draw[densely dotted] (EHP) -- (HHP) ;
19    \draw (EHP) -- (FHP) -- (HHP) ;
20  }
21  }

```

The animation :

A visual proof of Pythagore's theorem.