BeamToon

User's guide

University of Namur

BAYET
Anthony

Original Idea:
JACQUET
Jean-Marie

2019-2020

Contents

1 Overview

BeamToon is a project that is used to create animated and sonorised videos as well as powerpoints[1]. This project has 2 parts: an interpreter to display the BT projects and a language called "BTML" to describe them.
This installation part of this guide has been written for Windows 10 and Ubuntu 18.04. Therefore the steps described here are not ensured to work for any other versions of Windows or any other platforms such as MacOS.

1.1 Who can use it

After reading this document, anyone should be able to create a BeamToon project and to use it regardless of their initial skills.

1.2 Terms and conditions

This software, called "interpreter", is released under the GPLv3 license. For any other information consult the file "license.txt" at this url: github.com/UNamurCSFaculty/1920_INFOB3 18_Beam-Toon. Furthermore, BeamToon does not gather personal information of users.

2 Interpreter

The interpreter is the piece of software that will be used to read your BeamToon files and display their content.

2.1 Installation

To download the interpreter go to one of the url below, depending of your operating system.

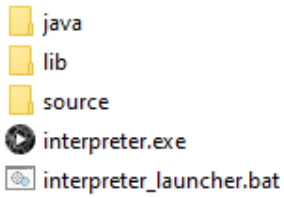
- Windows : github.com/UNamurCSFaculty/1920_INFOB318_Beam-Toon/releases/download/windows.1/application.windows64.zip

[1] BeamToon is actually able to do more than videos and powerpoints, for example interactive videos or even application prototypes. Therefore, throughout the rest of this guide we will talk about "BT projects" or "BeamToon projects" instead of videos or powerpoints.

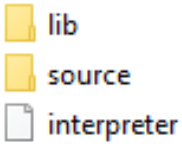
- Linux : github.com/UNamurCSFaculty/1920_INFOB318_Beam-Toon/releases/download/linux.1/application.linux64.zip

Once the download is over, extract the entire zip file.

Now, depending of your operating system you should end up with a folder containing this for Windows' users:



And this for Linux's users:



Now that you have installed the interpreter, the next sections are going to show you different ways to run a BeamToon project depending of your operating system.

2.2 Run (Windows 10)

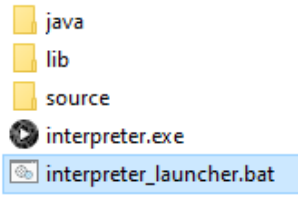
This section is describing 3 different ways to run a BeamToon project on Windows 10.

2.2.1 Locate the file from the interpreter

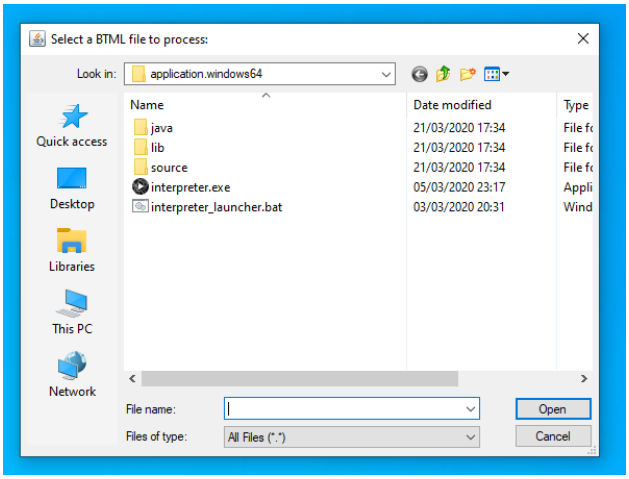
This method is about locating the main[2] file of your BeamToon project from the file selection tool built-in the interpreter.

In your interpreter folder, double-click on the file "interpreter_launcher.bat"

[2]cf. 3.1 Main file



This will open a file selection window like the one below.



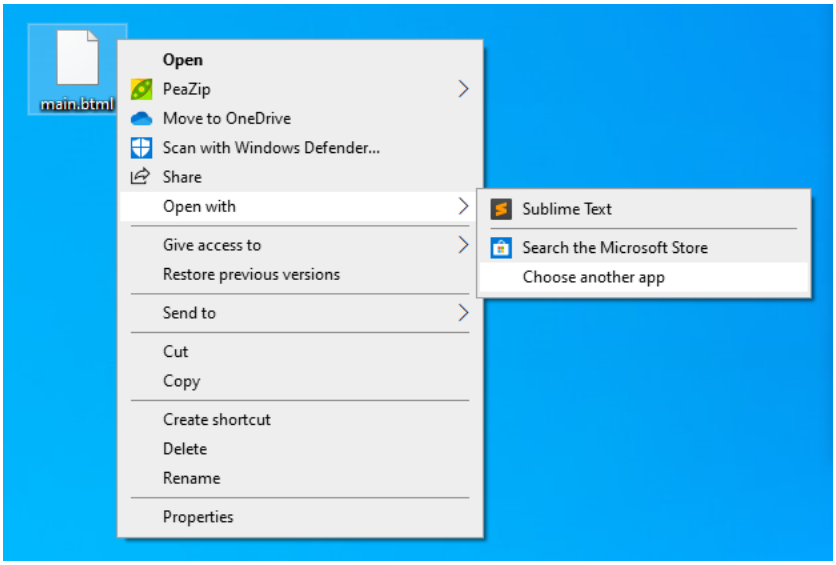
Then find the main file of the project to select it. And hit the ”Open” button.

2.2.2 Drag&drop the file on the interpreter

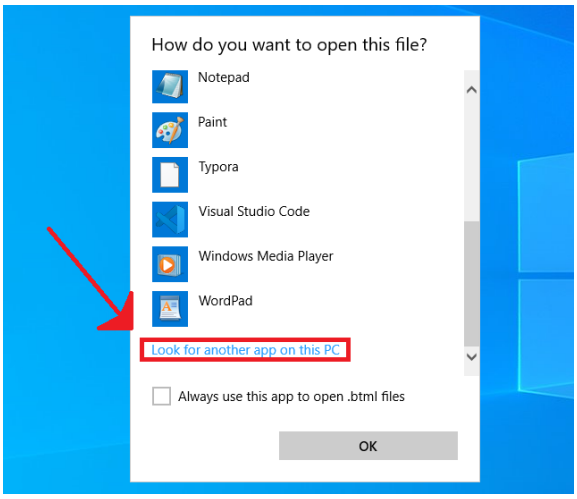
This method is the simplest one in term of number of step. You just have to drag the main file of the project on the ”interpreter_launcher.bat” file in your interpreter folder. And this will automatically launch the project.

2.2.3 ”Open with” the file with the interpreter

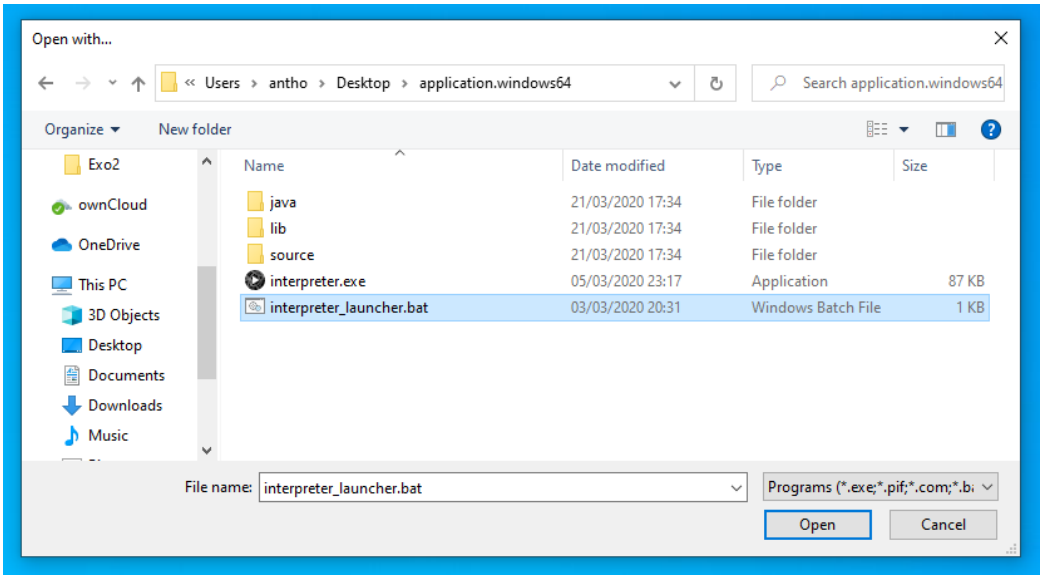
To run a project with this method, first right-click on your file ”project.BTML”, put your mouse on ”Open with”. This will open a sub-menu, in which, you click on ”Choose another app”.



A new window will pop up. Click on "More apps". Then scroll down till you see the "Look for another app on this PC" button and click on it.



This will open a window to let you localize the file named "interpreter_launcher.bat". If you have rigorously followed this installation section, the file should be located in the folder named "BeamToon_interpreter" on your desktop. Click on that file, then on the "Open" button.



2.3 Run (Linux)

To run a BeamToon project on Linux you can use the same method as specified in the ”Locate the file from the interpreter” and ”Drag&drop the file on the interpreter” subsection of the ”Run (Windows 10)” section.

3 Structure of a project

In this section, there are the descriptions of each part that compose a BeamToon project.

3.1 Main file

The main file is unique in a project. It is where most of the content that will be interpreted to be displayed is described in BeamToon Markup Language[3] (also called BTML). If something is not referred in the main file, then it will not be considered at all by the interpreter.

[3]cf. 4. BeamToon Markup Language

But in order to help users structuring their work, they can split their work in multiple files, called import files, and refer to them in the main one[4].

3.2 Import files

Import files are also BTML files, except that their goal is only to provide aliases[5] to the main file. Also they do not require a meta tag[6] to be used as an imported file.

3.3 Resources

These are simply external assets that are referred in the main file to make use of them. They can be files of 2 types: image[7] or sound[8].

4 BeamToon Markup Language (basis)

BeamToon Markup Language is also simply called ”BTML”. The syntax of this language has to be carefully respected by the users. Throughout this section we will see the different basic aspects of this language.

4.1 Generalities

The BTML structure is based on the XML norm. Thus, there are some generalities that are important to understand before talking about the BTML characteristics. The structure will be composed of elements. An element consists of a start-tag, an end-tag and a body between them.

```
<name_tag name_attr1="value" name_attr2="value">

</name_tag>
```

Figure 1: In this example the body is empty.

[4] cf. 4. Import
[5] cf. 5 Alias
[6] cf. 4.2.2 meta
[7] cf. 4.3.1 rectangle. Accepted extensions are: PNG, JPEG, JPG.
[8] cf. 4.2.3 slide. Accepted extensions are: MP3, WAV.

4.1.1 start-tag

The start-tag format is as described below:

```
<name_tag name_attr1="value" name_attr2="value">
```

”name_tag” is a keyword which is either built-in or defined by yourself[9]. ”name_attrX” is an attribute which has a value associated with it, that is next to the equal sign in between quotation marks. The attributes do not have to respect any order and if one is not recognized it will simply be ignored.

4.1.2 end-tag

The end-tag format is as described below:

```
<name_tag/>
```

where ”name_tag” must be the same as the corresponding start-tag.

4.1.3 body

The body of an element is either composed of other elements or is empty. If it is empty you can forget the end-tag. Then it looks like that:

```
<name_tag1 name_attr1="value" name_attr2="value"/>
```

otherwise if the body is composed of 2 elements you will get something like this:

```
<name_tag1 name_attr1="value" name_attr2="value">
    <name_tag2 name_attr1="value"/>
    <name_tag3/>
<name_tag1/>
```

For the rest of this guide, we will refer to ”name_tagX”, in this case, ”name_tag2” and ”name_tag3”, as children, as long as they are inside the first ”name_tagX”, in this case ”name_tag1” which will therefore be referred to as parent.

[9] cf. Alias section.

4.2 Structure

The structure of a BTML file is very strict. And its syntax has to be respected, otherwise the interpreter will not be able to run the project. Below are tags that are needed to create a BeamToon file.

4.2.1 root

The ”root” tag is necessary and it must be the parent of all. In other words, it must embed all the other elements. It has no attributes and it is unique.
Example:

```
<root>
    <name_tag attr1="value" attr2="value"/>
</root>
```

4.2.2 meta

The ”meta” tag is necessary and it is the direct child of the root tag. This tag describes the initial state of the project when it is executed by the interpreter. Therefore, it has 3 attributes:

- height: a numeric value of the initial height of the window.
- width: a numeric value of the initial width of the window.
- firstSlide: a word that is the name of the first slide[10].

Example:

```
<meta firstSlide="main" height="50" width="120"/>
```

Furthermore there can only be one ”meta” tag in a BT project.

[10] cf 4.2.3 slide

4.2.3 slide

The ”slide” tag is the heart of a BeamToon project, it is a direct child of the ”root” tag. This tag will be used to describe what will appear on the screen. You can have as many ”slide” elements as you want.

Its attributes:

- name: a word that will be used to refer to this slide. It must be unique in the main file.
- music (optional)[11]: the path[12] to the music that will start playing when the slide is displayed.
- stopSong (optional): its value is either ”true” or ”false”. If it is ”false” and the ”music” attribute is present, then the music will keep playing when the slide is changed. Otherwise it will stop directly after a new slide has been displayed.
- duration (optional): a numeric value (in millisecond) of how long the slide must stay on the screen. Afterwards if there is an existing slide name set as the value of the ”nextSlide” attribute, the corresponding slide will be displayed. If there is no value for ”nextSlide”, nothing happens.
- nextSlide (optional): name of the next slide that will be displayed if the current one stays on the screen longer than the value of the duration.

Example:

```
<slide name="nameThisSlide" toRedraw="true" duration="1500"
       nextSlide="nameNextSlide" music="musicFile.mp3" stopSong="false">
```

To describe the elements to display, you need to add them in the body of the slide. If you do not add any element, there will only be a grey screen with the name of the slide in the middle. These elements are described in the ”Visuals” Section.

[11](optional) means that the attribute is not needed to make the project run. Therefore, it means that any attribute that does not have this precision must be present, otherwise the project will not be understandable by the interpreter.

[12]The path can either be absolute or relative. The relative one will start from the main file of the BeamToon project.

4.3 Visuals

”Visuals” is the name of the syntax elements that will be used to describe how to display things on the screen. A ”visual” must be a child of either a ”slide” or another ”visual”.

They all have 4 attributes in common, which are:

- x: a numeric value representing the x coordinate of the upper-left corner of the visual relatively to its parent.
- y: a numeric value representing the y coordinate of the upper-left corner of the visual relatively to its parent.
- height: a number which is the height of the element.
- width: a number which is the width of the element.

For these attributes, if they are not written they will have the default value of 0 assigned.

At this moment there are 3 visual tags built-in: text, rectangle and shape.

4.3.1 text

The tag ”text” is used to display a text on the screen. In addition to the commons one, it has 2 attributes:

- text: text that will be displayed on the screen
- police (optional): numeric value of the size of the text. If you omit it, it will take the default value of 16.

Example:

```
<text text="Text content here" width="30" x="80" y="20" police="16"/>
```

4.3.2 rectangle

This element is used to display a filled rectangle. And its only additional attribute is:

- path (optional): the path of the image, that will be displayed as a rectangle.

Examples, the second with the path attribute and the first without:

```
<rectangle x="20" y="10" height="100" width="30"/>
<rectangle path="data/img.jpg" height="100" width="100" x="0" y="0"/>
```

4.3.3 shape

This element is used to represent any shape. To do it the body of this element is composed of 2 type of child: vertex and bezierVertex. You can see a vertex as a point and the first child of a shape must always be a vertex. Then the shape will be displayed by linking the vertex and bezierVertex together depending of their type. If there are 2 simple vertexes, it will draw a line between them.
If the second one is a bezierVertex, it will draw a bezier curve with (x1;y1) and (x2;y2) as coordinates of the control points, (x3;y3) as coordinates of the end point. And for the start point there are 2 scenarios:

- A vertex is placed right before the bezierVertex. Then it will be the start point.
- A bezierVertex is placed right before the bezierVertex. Then the start point of the second bezierVertex will have the same value as the (x3;y3) of the first bezierVertex.

Examples:

```
<shape x="100" y="100">
    <vertex x="10" y="10"/>
    <vertex x="10" y="100"/>
    <vertex x="100" y="100"/>
    <vertex x="100" y="10"/>
</shape>
```

```
<shape x="100" y="100">
    <vertex x="30" y="20"/>
    <bezierVertex x1="80" x2="80" x3="30" y1="0" y2="75" y3="75"/>
    <bezierVertex x1="50" x2="60" x3="30" y1="80" y2="25" y3="20"/>
</shape>
```

4.4 Visuals' attribute

Here are explained 2 special tags that are used to add some settings to visuals. These are "color" and "clickActionGoTo".

4.4.1 color

The color tag is used to define the color of its parent using the rgb scale. It has no attribute but must have 3 children, which each have 1 attribute named "value":

- (child) red: its value attribute describes the quantity of red in the rgb scale of that color.
- (child) green: its value attribute describes the quantity of green in the rgb scale of that color.
- (child) blue: its value attribute describes the quantity of blue in the rgb scale of that color.

Examples:

```
<color>
    <red value="100"/>
    <green value="30"/>
    <blue value="30"/>
</color>
```

4.4.2 clickActionGoTo

A clickActionGoTo element must describe which slide will appear when its parent is clicked on[13]. Its only attribute is "name" whose value is the name of the next

[13]To determine when an element is clicked on the interpreter imagine a rectangle with the same x, y, height and width value of the real element. And when a click occurred it computes if the mouse is in this rectangle or not, and this correspond to the state of being clicked or not.

slide. Therefore there must be a slide with the name in the same project otherwise the program will stop working.

Examples:

```
<clickActionGoTo name="main"/>
```

5 BeamToon Markup Language (advanced)

In this section, we will see advanced mechanics of the BTML. Thus, be sure to feel ease with the basis before continuing.

5.1 Alias

The Alias mechanic is the most powerful feature of BeamToon. It allows you to abstract part of your creation to reuse them in it and in other projects. In other words it consists of the creation of new tags.

Practically an alias has 2 parts called: signature and definition.

5.1.1 Signature

The signature is always the first part in an alias, in it you are going to describe how you can use ”redColor” in your project. This part of the alias is a tag with a tag name[14] and some attributes but no body. There must be a value for each attribute, because they are their default value.

5.1.2 Definition

The definition must always come in second place in the body of an alias. And it represents how this new tag must be interpreted. Practically it is a tag, and this tag can have a body with as much child as you want. But the value of each attribute of the parent tag or any child must be an attribute name that is in the signature.

[14] It is strongly recommended that the tag name is unique for the entire project and different than the predefined tags of BTML. Except you know very well what you are doing.

5.1.3 Example

Situation: we have this BTML code:

```
<shape x="100" y="100">
    <vertex x="10" y="10"/>
    <vertex x="10" y="100"/>
    <vertex x="100" y="100"/>
    <vertex x="100" y="10"/>

    <color>
        <red value="250"/>
        <green value="0"/>
        <blue value="0"/>
    </color>
</shape>

<rectangle x="200" y="200" width="50" height="50">

    <color>
        <red value="250"/>
        <green value="0"/>
        <blue value="0"/>
    </color>
</rectangle>
```

But having at 2 different places the same piece of BTML code makes us sick. Then we want to create an alias to abstract the red color. And we do this alias:

```
<alias>
    <redColor red="250" green="0" blue="0"/>

    <color>
        <red value="red"/>
        <green value="green"/>
        <blue value="blue"/>
    </color>
</alias>
```

Therefore we make use of it in our original code and we end up with a code like that:

```
<shape x="100" y="100">
    <vertex x="10" y="10"/>
    <vertex x="10" y="100"/>
    <vertex x="100" y="100"/>
    <vertex x="100" y="10"/>
    <redColor/>
</shape>

<rectangle x="200" y="200" width="50" height="50">
    <redColor/>
</rectangle>
```

Notes: we could have set another value to any attribute of redColor when referring to it in ur code to make it blue for example.

5.2 Import

The goal of the import tag is to use the aliases of an external file in your main file. Thus when the interpreter starts its work it will at first check all import tags in your code and apply the aliases in your main file.

The usage of this tag is as it is shown below:

```
<import path="toimport.btml"/>
```

5.3 ToRedraw

”ToRedraw” is a special attribute common for the tags: rectangle, text, shape and slide. It is a boolean and if it is set at ”true” the element having the attribute will only draw once when the slide is displayed. To understand this you must at first know how the interpreter displays elements on the screen. When the interpreter is asked to display a slide it will at first draw a background depending of the color of the slide. Then it will display elements one by one in the order as they appear in the code. And then the interpreter will do this operation as often as it can. The problem is that when you want to display fat images it will slow down the program. Therefore to solve that problem you can set the toRedraw attribute of the slide and the image to ”false”. And if there is no other element overlapping on the image it will be displayed and kept on the screen will not slowing down your BeamToon project.

5.4 Child attribute

We have seen in the section 4.1.1 that the attributes of an element are put in its start-tag. But some attributes have actually another way to be represented. This is done by creating a child with the name of the attribute as the one of the tag, and with a single attribute ”value” that is the value of the attribute. The attributes that have this property are: height, width[15], x, y.

Example:

```
<rectangle y="10" height="100">
    <x value="20"/>
    <width value="30"/>
</rectangle>
```

But the main objective of this feature is to make your BeamToon project dynamic. That is what the next section is about.

5.4.1 Animation

To animate your project there is a built-in feature naturally called ”Animation”. In BeamToon an animation will change the value of an attribute linearly between set values.

Practically, you add an animation child to the ”Child attribute” you want to animate. This animation tag has 3 attributes:

- startValue: the numeric value being the initial one of the animated attribute.
- infinite (optionnal): it is a boolean and if set to true it will replay the animation when it is over. Its default value is false.
- boomerang (optionnal): it is a boolean and if set to true it will double your animation but when the original is ended, it will replay it but backward. Its default value is false.

And that animation tag needs children to create the actual animation. Those children are keyframe tags and there must be at least one in the body of the animation. This type of tag has 2 attributes:

[15]height and width as children will not work for the meta tag

- value: the numeric value that the animated attribute will have at that frame.
- timing: the numeric value representing the moment (in millisecond) when the animated attribute must reach the value set in this keyframe. This attribute must be greater than 0.

Each keyframe of an animation must have a smaller timing value than the following ones. Furthermore the initial timing value of an animation is 0 and an animation starts when the slide parent is displayed.

Example:

```
<height>
    <animation startValue="50">
        <keyframe value="50" timing="500"/>
        <keyframe value="200" timing="1000"/>
    </animation>
</height>
```

6 Frequently asked questions (F.A.Q.)

6.1 Can I install the interpreter on MacOS

No you cannot. At this moment there is no available version for MacOS.

6.2 My BeamToon project does not work, why

It can be for multiple reasons, but common fixes are:

- Ensure that your BT file is correctly written, and has a correct syntax, as described in the BTML Section[16].
- Check if the total size of your files (BT file + images/sounds), is less than 2Gb.

[16] cf. 4 BeamToon Markup Language (basis) and 5 BeamToon Markup Language (advanced)