VRSpace.org - Bug #125

gltf characters with babylon 5

06/01/2022 07:07 PM - Josip Almasi

Status:	Closed	**Start date:**	06/01/2022
Priority:	Normal	**Due date:**	
Assignee:	Josip Almasi	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Babylon 5 introduced undocumented breaking changes to gltf character structure.
Some explanations are found in following babylon forum threads:
https://forum.babylonjs.com/t/inconsistent-gltf-bone-transformations/29358
https://forum.babylonjs.com/t/gltf-parsing-is-creating-doubles-and-reparenting-meshes/30079/18

Bottom line, vrspace client heavily relies on Bone.getAbsolutePosition() which now works differently.
Differences include poorly documented parameters, but also need to call computeWorldMatrix(true) on TransformNode associated with the bone after e.g. resizing.

One critical fix for head position was applied right away:
https://github.com/jalmasi/vrspace/commit/d1c366159d5b7a019ae7ad72da4dc54fda50cb8a
But this quick fix breaks down everything in XR, as it calls Scene.render() to ensure that getAbsolutePosition() retrieves valid coordinates, and calling Scene.render() in XR is inheritly unsafe.
First thing to fix here is get rid of Scene.render, and use getAbsolutePosition() of head Bone rather than TransformNode, calling computeWorldMatrix if appropriate, as appropriate.

Second thing is to figure out is the orientation of arm and leg bones, required for IK. I.e. around which local axis to apply rotation to move an arm to desired direction.
Avatar class tries to do that in guessRotation and tryRotation methods, by applying rotation in every direction and measuring bone coordinates.
This is definitely the wrong way to do it. This can and should be calculated, I just don't know how.
It was good enough as it's done only once per character, but it doesn't work any longer.
Testbed: https://www.vrspace.org/babylon/avatar-test.html
For example lisa, tiffany, blake_fbx, legendary_robot all rotate in all the wrong ways.

EDIT there's another one:
https://forum.babylonjs.com/t/thininstanceadd-incorrect-z-position-on-imported-models-vs-babylon-primitives/33124

Related issues:

Related to Support #115: babylon 5.0.0	**Closed**	**03/31/2022**
Related to Bug #126: Crash after loading gltf	**Closed**	**06/03/2022**

History

#1 - 06/01/2022 07:08 PM - Josip Almasi

- Related to Support #115: babylon 5.0.0 added

#2 - 06/03/2022 02:49 AM - Vander Dias

- Related to Bug #126: Crash after loading gltf added

#3 - 06/04/2022 03:15 AM - Vander Dias

Babylon.js skeleton handling is very finicky.

I tested a few models on the mesh viewer and they're ok. Touching bones is known to be problematic on BJS, in special with skeletons which have different hierarchies.

Some of the current models have additional bones, which might be one of the reasons for the bugs (?) - need to investigate more in that regard.

I am now experimenting a few things and will have more info soon.

#4 - 06/04/2022 12:21 PM - Josip Almasi

Nah, additional bones are ignored, e.g. buttock, breasts. A number of developers asked how I identify bones so I wrote it in the paper: https://www.researchgate.net/publication/356987355_TOWARDS_AUTOMATIC_SKELETON_RECOGNITION_OF_HUMANOID_3D_CHARACTER I think it's out of scope here. Sure, I may be wrong :)

Note that you may enable BoneAxesViewer to see the orientation, it's there already but commented out. I'd say we simply fail to determine bone orientation in guessRotation.
'Simply' ;)

#5 - 06/06/2022 03:01 AM - Vander Dias

Yes, today I studied this part of your code and found that you scan the hierarchy by looking their partial names.

Did further tests with the tracking and you're right, the problem is related to bone orientation.

Thanks for pointing guessRotation, will be focusing more into that.

#6 - 06/06/2022 03:24 AM - Vander Dias

As food for thought, I was recently experimenting with MediaPipe (here: https://forum.babylonjs.com/t/realtime-3d-face-capture/30020 and here: https://forum.babylonjs.com/t/wip-full-body-3d-capture/30149).

My plan was to approach the tracking by finding only angles between bones, then rotating the bjs bones to mimic the desired pose.

At this moment I did not read enough your code, but bringing my own strategy to the table, I believe that using only rotations might work well as we could work with relative rotations which are not problematic.

You know, ok we got the hand and orientation from Quest. Perfect, now we can guess arm and shoulder (a triangle, you see) by doing a simple ik calculation.

Then, instead of trying to set positions, we only find the angles between those three bones -- again, they form a triangle and we can take advantage of that.

After finding the angles, we set those angles in bjs, but not the absolute angles, but the relative angles starting from the rest pose angles.

Hope this makes sense.

#7 - 06/06/2022 12:44 PM - Josip Almasi

It does make sense, and that's much like what we're doing now.
The angle of the controller is used for arm angle - if controller is available. That's part of the code in reachFor method in if (arm.pointerQuat) block, tries to figure out the plane we'll bend arm on, and bendArm is called last. This implementation works fine as long as controller remains below shoulders, then it goes off :)
E.g. like here https://www.youtube.com/watch?v=05Tmra7nSTY&t=245s (also watch next minute or so, as it explains EventRecorder that I used to debug avatars)
There's also another debug tool in the code commented out, drawVector calls.

So, what could possibly go wrong?
- there's no controller - PC or mobile user or a bot
- avatar is 200 times bigger than human
- axis of rotation is swapped - typically Z and Y but you never know
- rest pose, are we talking T or A pose or...? ;)

Thus, a big amount of essentially boilerplate code, or should I say boilerplate math, to map one coordinate system to another one.
Like, you know position of the controller, first thing you need is to find what it is in local coordinate system of the shoulder.
Like:

```
    // calc target pos in coordinate system of character
    var target = new BABYLON.Vector3(t.x, t.y, t.z).subtract(totalPos);
    // CHECKME: probable bug, possibly related to worldQuat
    target.rotateByQuaternionToRef(rootQuatInv,target);
```

```
    // calc target vectors in local coordinate system of the arm
    var targetVector = target.subtract(armPos).subtract(totalPos);
    targetVector.rotateByQuaternionToRef(worldQuatInv,targetVector);
```

#8 - 06/07/2022 03:43 AM - Vander Dias

I was meaning TPose.

Usually, as you certainly know, skeletal animations start from the TPose (or APose which is less common, afaik).

So, did you at some point had all those models -- those without TPose -- working well? Then a babylon.js update broke the animation system? Or those never worked well even before?

I ask beause I find this unusual to use models which don't have an initial TPose. If I'm not misled, those require absolute rotations which is what I suppose you use to put them at a known pose.

And that is what I was referring to. If we have a model in a well known initial pose -- let's say TPose -- we can then use relative rotations, which for quaternions is not a problem at all.

Please note that while I post this, I'm also studying your code and checking models etc. I will reach something soon.

#9 - 06/07/2022 03:45 AM - Vander Dias

What I mean with the above is that if you use random models in random poses, you will always find some that won't work properly. You might adapt your code and make them work, but then new ones might break it again and so on. Using absolute rotations won't always work. That is why TPose/APose were invented, I guess.

Please enlighten me if I'm talking garbage here.

#10 - 06/07/2022 03:48 AM - Vander Dias

So, now spamming a bit, sorry -- I think that the ideal would be to make sure that all models are initially in TPose. That is quite some work, open them one by one in Blender and put them in TPose... but that would make the system much more reliable, because we could use relative quaternion rotations and they would always work well.

#11 - 06/07/2022 03:50 AM - Vander Dias

Or maybe that is exactly a diffential that you want? To be able to animate any model, from any initial pose? Problem is, they will also have random rotations, seems very hard to put them in a known, reliable pose from there.

Please enlighten me.

#12 - 06/07/2022 04:08 AM - Vander Dias

One good thing is that all of the models I checked until now had either APose or TPose (which I prefer).

So, bjs has the skeleton.returnToRest() which puts the skeleton to its initial pose.

Now, if we at least know when a model has APose or TPose, we can also know which relative rotations we can use. And if I'm not very very mistaken (I believe not)... that would make the system much more reliable.

#13 - 06/07/2022 04:10 AM - Vander Dias

I am repeating "relative" so much because for quaternions, if we use relative rotations from a known pose, it does not matter which is the current rotation, it only matters the axis of rotation. And even if the axis is swapped etc, knowing the initial pose allows us to detect and fix that reliably.

Now, I need to understand a bit more about your code. I will try to shutup and come back when I'm ready to make changes.

Please forgive the spamming. =)

#14 - 06/07/2022 04:13 AM - Vander Dias

Just found out that you actually call returnToRest() at avatar.js.

But some models have APose, others have TPose, do you handle that already?

#15 - 06/07/2022 04:54 AM - Vander Dias

Tomorrow I will redo this test with all models and take better note, but one thing that I noticed today was:

- All of the models (tested only random females for now) which had initial TPose worked;
- Most but not all of the models in APose worked, but some not.

In special: blue, caroideira, valora, vanille did not work, and all them had APose as rest.

#16 - 06/07/2022 10:09 AM - Josip Almasi

Vander Dias wrote:

> So, did you at some point had all those models -- those without TPose -- working well? Then a babylon.js update broke the animation system? Or those never worked well even before?

All these characters worked equally well before babylon 5, for 2 years or so. I've come up with the system that works for 96% rigged humanoid characters, in any rest pose. Ones that did not work had their joint names in Spanish :)
I suspect last working version is Babylon.js v5.0.0-beta.11 - I haven't tried it myself, but that's the one used by framevr.io. (this skeleton breaking change affected everyone)

For example of malfunctioning avatar in T-pose see lisa. An example of a functional avatar in custom pose see dark_astronaut.

Pose is important in VR only, when we track controller rotation. Then the angle between the arm and the body is set to track the angle of the controller. That's part of reachFor method after
// vector pointing down in local space:

#17 - 06/08/2022 02:13 AM - Vander Dias

That is very cool, something to be really proud of!

Today I have been debugging the code. I enabled the BoneAxesViewer for all bones (created an array and added one on each guessRotation call).

I used the "anime-blue" as the test model. Interesting enough, it does not show all bones. Apparently, the code is not scanning the entire meshes array to find all the bones. Also, as we already know, the bones which were added were in wrong places. I don't know yet why that happens.

So, to check if it was a bug in Babylon.js itself, I copied this model to one of my ongoing projects, where I use animated humanoids. I then did a similar thing, added BoneAxesViewer to each scanned bone.

Here is the image:

mdhIBkl
Then I went to the "lisa" model. That one behaved exactly as in your code. All bones among the legs, at ground. Totally wrong. I suspect though that that is a different problem -- scale, maybe. I opened this model in Blender and Blender was not even able to render the model properly. It was all screwed up. It seems to be really huge in scale.

Ok, I don't have a solution yet, so I'm just thinking loud, sorry.

For now my two somewhat blind guesses are:

-- Some models are not being scanned entirely (bones being missed);
-- Some models have weird scales either for the bones, or for the meshes, or both.

Will continue investigating...

#18 - 06/08/2022 02:25 AM - Vander Dias

The above did not show the image.

Here it is again:

https://imgur.com/a/mdhIBkl

#19 - 06/08/2022 02:30 AM - Vander Dias

Oh my God, now I see, you don't try to scan all bones anyway... only the ones that are important to you. Sorry for that.

Anyway, let me investigate further. This is intriguing. =)

#20 - 06/08/2022 02:37 AM - Vander Dias

One thing I noticed: on anime-blue, the bones are in the right place, but rotated 90 degrees in relation to the model!

https://imgur.com/a/mffRzWD

Ok, so I'll stop spamming and will proceed investigating. you'll have to trust that I'm working on it then. See you later, when I have a more solid progress.

#21 - 06/08/2022 08:37 AM - Josip Almasi

Interesting! The answer may be in https://forum.babylonjs.com/t/inconsistent-gltf-bone-transformations/29358/4

#22 - 06/09/2022 04:51 AM - Vander Dias

Good news: I got the anim-blue working nicely!

Bad news: I needed to open it in Blender and Object -> Apply -> All Transforms on the four first transforms of its hierarchy.

I can detail how I found this but maybe you're not interested for now.

I am too tired at this moment but tomorrow I will open and normalize the others which are not working to see if they get fixed as well.

Now, there is another bug which is related to the legs, they bend when I look to the ground -- but I believe that is unrelated? Will investigate that soon.

#23 - 06/09/2022 10:13 AM - Josip Almasi

Uh, that kinda defeats the purpose. Can we do that in the code rather than in blender?
Now that's an interesting find definitely, I haven't found that these characters have anything in common. Please push your modifications into a branch so I can see for myself.

Ah in VR you look down, I see what you mean. Not really a bug, it's just how things work.
XRCamera realWorldHeight() changes as you look up and down. We may apply correction depending on angle... say alpha is looking up down angle, alpha = 0 when looking straight forward. Correction is k*sin(alpha), where k is about the size of human head, or maybe distance from neck to the eyes. Whatever, tweak it till it works for you ;)

#24 - 06/09/2022 01:30 PM - Vander Dias

- File avatar.js added

- File anime-blue.zip added

> git push -u origin task125

ERROR: Permission to jalmasi/vrspace.git denied to imerso.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

==> I attached anime-blue and avatar.js here

#25 - 06/09/2022 01:37 PM - Vander Dias

- "Can we do that in the code rather than in blender?"

Certainly, although I can't say how much time/effort it will be.

Apply -> All All Transforms normalizes everything, like, applies translation/rotation/scale in-place, normalizing translation=0,0,0 rotation=0,0,0 and scale=1,1,1. Maybe it's be quick to do the same thing in bjs, maybe not, I don't know right now.

#26 - 06/09/2022 02:04 PM - Josip Almasi

Vander Dias wrote:

> > git push -u origin task125
>
> ERROR: Permission to jalmasi/vrspace.git denied to imerso.
> fatal: Could not read from remote repository.

Read.
Huh, how did you clone the repo? If you did over ssh you should be able to push. Or at least wouldn't get read error, it's public repo.
We need to get it solved anyway.

#27 - 06/09/2022 02:32 PM - Josip Almasi

Vander Dias wrote:

> - "Can we do that in the code rather than in blender?"
>
> Certainly, although I can't say how much time/effort it will be.

Well that's why I need you, to fix the code :)

There's one thing that I didn't mention earlier, as I didn't want to lead you on the wrong track.
I've spent a few days at this, the result is in avatar-debug branch. It still doesn't work, and it went in the wrong direction.
So what I've picked on babylon forum is
1) Bone.getAbsolutePosition(transformNode)
2) transformNode.computeWorldMatrix(true)
But the question is, which transformNode to use. I thought I had it figured out, but never received any confirmation on babylon forum.
So I had everything reworked to use getAbsolutePosition(transformNode) instead of getAbsolutePosition(), with mixed success.
But what I ended up with requires transformNode.computeWorldMatrix(true) after any manipulation.
Forcing computation on every avatar for every movement is obviously wrong, so forget it.

So back to your findings.

It is known that direct manipulation requires transformNode.computeWorldMatrix(true), but which transform node, that's not known.
Bone.getAbsolutePosition() is now an absolute misnomer :) I had a look at the code and I'm not clear which coordinate system it uses, but one thing sure, it's not absolute.
If you look very carefully at how tryRotation() works, I mean debug

```
    bone.setRotationQuaternion(quat.multiply(rotated));
    bone.computeAbsoluteTransforms();
    var newPos = target.getAbsolutePosition();
```

you'll see that newPos doesn't really work for these avatars.
But this part of code is executed only once, when character loads, so forcing recalculation is cheap enough to be useful.
And this is reasonably small piece of code that may be reproduced in babylon playground, and file bug report.
Small, like
- load avatar
- rotate an arm
- get coordinates
Hehe sounds soooo easy ;)

#28 - 06/09/2022 03:02 PM - Vander Dias

I used ssh. Just guessing: maybe I don't have permission to create a branch?

Thank you for yet another explanation. =) I will try that.

I believe we are approaching a solution.

#29 - 06/09/2022 04:03 PM - Vander Dias

Have made an important progress here, will update later today!

#30 - 06/09/2022 04:16 PM - Vander Dias

Ok, so now I found out that by just clearing rotations on the first nodes from the hierarchy makes anime-blue work nicely from code.

Problem is that it makes the others have the problem it had before.

What I need now is a way to detect when it's needed or not.

#31 - 06/09/2022 08:07 PM - Vander Dias

https://github.com/jalmasi/vrspace/tree/task125

#32 - 06/09/2022 08:55 PM - Vander Dias

Correction: "works nicely" in the sense that the bones go to the right places, but in VR most rotations are messed up yet.

#33 - 06/09/2022 11:20 PM - Vander Dias

I made a playground:

https://playground.babylonjs.com/#H925CH#2

You can see there that the model loads correctly there.

There is something incompatible that is being done to the skeleton which makes it screw up.

Up to this point, I have seen two cases: one happens on anime-blue, where the bones are 90º rotated. The other case is on the model above, where bones are all close to each other between the legs.

I implemented a "detector" for the first case and successfully rotate the skeleton to put it where we expect. It is pushed to the branch "task125".

The second case I am still investigating. But there is certainly something "breaking" -- yes, I know, not really a bug per se but something that Babylon does not like.

#34 - 06/09/2022 11:21 PM - Vander Dias

No way to edit posts here?

The PG above is the old version. Here is the correct playground link with Mannequin:

https://playground.babylonjs.com/#H925CH#9

#35 - 06/09/2022 11:22 PM - Vander Dias

Please use the arrow keys to get close to it, I made the axes small so we can see all them without cluttering.

#36 - 06/09/2022 11:55 PM - Vander Dias

Now trying to reproduce on PG:

https://playground.babylonjs.com/#H925CH#10

The BoneAxesViewer definitely behaves weird. I want to have more solid proof to open a bug report, though.

#37 - 06/09/2022 11:58 PM - Vander Dias

Problem is, in the last PG above, it does rotate properly... BoneAxes is wrong, but the mesh itself is bending as expected. Annnd, if we don't rotate anything, it shows perfectly, with BoneAxes and all.

???

#38 - 06/10/2022 12:22 AM - Vander Dias

Bug Report:

https://forum.babylonjs.com/t/boneaxesviewer-does-not-follow-bones/31223

#39 - 06/10/2022 12:23 AM - Josip Almasi

Sure you can edit post, click on the edit icon above the post :)
Dude, I'm so glad that you've so eager, but remember that you need to sleep sometimes!
And so do I - talk tomorrow :)

#40 - 06/10/2022 01:04 PM - Vander Dias

Sorry, I don't see the "edit" button, I see only the "quote" button. =)

#41 - 06/10/2022 06:30 PM - Josip Almasi

Vander Dias wrote:

> Sorry, I don't see the "edit" button, I see only the "quote" button. =)

That's just silly, seems you have to be the project manager to edit your own post :)
Now you are.

#42 - 06/10/2022 09:39 PM - Vander Dias

https://playground.babylonjs.com/#H925CH#21
(only one model loaded at a time now)

Please note that the animation is procedural, not from the model.

I chose those models because none of them currently work in VRSpace with Bjs 5 -- but if you check that PG, all them are animating equally well.

I stand by the initial theory that using relative rotations is more reliable. You see, without forcing rotations or positions, it works smoothly.

Not saying that the current implementation is wrong , it's just that it relies on a perfect skeletal handling by the engine, and most engines are pretty much buggy in that regard.

Yes, I know that you had it working before! Yes, right, but if we switch to a less intrusive approach (relative rotations everywhere) it will become more future proof.

If you don't mind, I'll keep improving the PG until it finds the rotation axes (which will be probably your current way because on preliminar tests it works perfectly already) and then puts the model in T-Pose, using only relative rotations from the initial loaded pose.

Then we find a way to integrate with the main code.

What do you think?

#43 - 06/10/2022 10:18 PM - Josip Almasi

Uh, I'm not sure what to think really.
If we have to rotating transform nodes rather than bones, so be it. At least that's a straightforward approach, that we can do.
On the other hand, I'm not sure if we can figure out bone orientation that way, guessRotation style.
One thing that we can't have is everything handled in render loop :) But it get it, this is just an example.
So I think you keep at it, I'll have another try at guessRotation over the weekend, using getTransformNode().rotate(). Who knows, maybe Bone.getAbsolutePositon() (sigh) provides useful information that way.

#44 - 06/10/2022 10:23 PM - Vander Dias

Only the rotations are in the render loop, please check again.

I believe that it will work similar to what is done currently, except that in this new way we avoid touching the skeleton too much, because it screws up when we do that. We can still find angles, then relatively rotate to put the model on a known pose, then from there rotate it to any desired pose, using only relative rotations.

https://playground.babylonjs.com/#H925CH#27

The above gets the transform node on initialization only. I also added XR so it runs on Quest as well.

#45 - 06/11/2022 01:33 AM - Vander Dias

But let's forget all that, I'll continue trying on guessRotation() and tryRotation(), must be something really obvious to get them working again.

#46 - 06/11/2022 01:51 PM - Josip Almasi

Vander Dias wrote:

> But let's forget all that, I'll continue trying on guessRotation() and tryRotation(), must be something really obvious to get them working again.

Hehe well obviously Bone.getAboslutePosition() does not work :)
Not that obvious is that IK can't be done without positions of hand, feet, head. Whether Bone or TransformNode, we must have some position; doesn't have to be very accurate though.
And the opposite of obvious, quite obscure in fact, is how to get any position and orientation, to any degree of accuracy, in global (scene) coordinate system. Not even BoneAxesViewer does that.

So let's not forget the playground just yet.
Can you use it to figure out position of hands, i.e. paint a sphere there or anything that makes sense?
What exactly is this used for:
container.skeletons[0].bones[b].getTransformNode().rotationQuaternion = container.skeletons[0].bones[b].rotation.toQuaternion();
?
I don't see any difference.
(debug viewers don't display for vanille, 3)

#47 - 06/11/2022 02:14 PM - Vander Dias

Have an emergency here.

On both my Quests (1 and 2), VRSpace is not rendering properly anymore, not even the original website vrspace.org!

Not sure if this was an Oculus update or Babylon, but not even the vrspace.org is rendering properly anymore here.

Could you please open vrspace.org or your local copy on Quest and check that?

Thanks.

#48 - 06/11/2022 02:20 PM - Vander Dias

"Can you use it to figure out position of hands, i.e. paint a sphere there or anything that makes sense?"

I believe so. In fact, your guessRotation() DOES WORK properly, **if** you don't touch the skeleton in some way on initialization. There is something on your initialization which breaks the skeleton (and up to now I could not find what exactly it is that breaks). Not saying it's wrong, it's just that Babylon does not like this something you do after loading the model.

I tested guessRotation() on a separate project like the above PG, and IT WORKED. Today I'll put guessRotation() on the PG and update it.

"What exactly is this used for: container.skeletons0.bones[b].getTransformNode().rotationQuaternion = container.skeletons0.bones[b].rotation.toQuaternion();"

There is a weird behavior on Babylon which is, if you use "rotation" (euler angles) it clears the rotationQuaternion (quaternion) and uses eules angles from there.

So, the line above is: rotationQuaternion is the quaternion, and rotation.toQuaternion() is reading the euler angles and then converting it to a quaternion.

#49 - 06/11/2022 02:24 PM - Josip Almasi

Right. Looks like Babylon 5.10.0 is out, and broke XR completely :)
In my case it is rendering something but images are out of sync.

#50 - 06/11/2022 02:30 PM - Vander Dias

Unbelievable!

~~Would you consider forcing a pre-5.0 version and wait until 5.x is stable enough? If I'm not mistaken, their cdn allows to pull previous versions. Edit: here~~ https://cdnjs.com/libraries/babylonjs

Other experiences are working, so there is something on engine initialization breaking it. Will try to find what is causing that.

#51 - 06/11/2022 02:56 PM - Josip Almasi

Vander Dias wrote:

> I believe so. In fact, your guessRotation() DOES WORK properly, **if** you don't touch the skeleton in some way on initialization. There is something on your initialization which breaks the skeleton (and up to now I could not find what exactly it is that breaks). Not saying it's wrong, it's just that Babylon does not like this something you do after loading the model.
>
> I tested guessRotation() on a separate project like the above PG, and IT WORKED. Today I'll put guessRotation() on the PG and update it.

That would be rescaling the model. It's done twice, in _processContainer().
First based on bounding box, general case for avatars that don't have skeletons, e.g. this dolphin. (Separation of Concerns needs to be applied here, but that's another issue)
And then the resize() is called, that sets avatar size to one specified in this.userHeight property.
Now that I'm looking at it, it doesn't enforce matrix recalculation after scaling, so that's the good candidate for what may go wrong.
And now that I'm playing with it and trying to enforce it, it turns out that
this.rootMesh.computeWorldMatrix(true);
this.skeleton.computeAbsoluteTransforms(true);
are not enough,
this.scene.render();
has to be called :)
And it doesn't seem to influence guessRotation() in any way.

#52 - 06/11/2022 03:00 PM - Josip Almasi

Vander Dias wrote:

> Unbelievable!
>
> ~~Would you consider forcing a pre-5.0 version and wait until 5.x is stable enough? If I'm not mistaken, their cdn allows to pull previous versions.~~

If I knew about this, I would have done it already :)
But it may not be that easy, we're also using dynamic terrain, grid material, possibly something else.
Let's not give up just yet, they usually fix catastrophes fast.

#53 - 06/11/2022 03:01 PM - Vander Dias

Maybe using a setTimeout() at some place to allow the engine to render a few frames, and then proceeding with the scaling, recompute etc could solve that.

Forcing render() on init breaks Quest.

Edit: well, unfortunately guessRotation() is not always working on the PG test.
https://playground.babylonjs.com/#H925CH#28

#54 - 06/11/2022 04:36 PM - Vander Dias

GOOD NEWS!

Using rotate with BABYLON.Space.WORLD inside tryRotation() made it detect front and side axes on all models.

Needs some work yet but it seems to be very very close to a solution now. Please note that the following PG uses guessRotation() and tryRotation() to find axes.

https://playground.babylonjs.com/#H925CH#31

```
tryRotation(bone, axis, angle) {
  var target = bone.children[0];
  var original = bone.getRotationQuaternion();
  var oldPos = target.getAbsolutePosition();
  bone.rotate(axis, angle, BABYLON.Space.WORLD);
  bone.computeAbsoluteTransforms();
  var newPos = target.getAbsolutePosition();
  bone.setRotationQuaternion(original);
  bone.computeAbsoluteTransforms();
  var ret = newPos.subtract(oldPos);
  return ret;
}
```

Only problem now is that I can't test on VRSpace in VR mode, because of the new BJS version. Will see if I can fix that thing so that we can test again.

#55 - 06/12/2022 01:00 PM - Josip Almasi

That is good news indeed. Looking back, I can't recall why I never used rotate() method. I only recently became aware of it, maybe it didn't even exist back when I implemented this first.
OK I'll try to play with it later today.

#56 - 06/13/2022 09:25 PM - Josip Almasi

OK I'm reasonably sure there's nothing wrong with tryRotation() by now :)
Bone.rotate and bone.rotationQuaternion work differently, but they both do seem to work.
I've confirmed it by hardcoding axes in guessArmsRotations() - that's where it's called from. E.g.
this.body.leftArm.sideAxis = {axis:BABYLON.Axis.Z, sign:-1}
And anime blue behaves exactly the same no matter what axis and sign are :)

That brings us to next suspect, reachFor() method. It first figures out current arm position like
var armPos = upperArm.getAbsolutePosition().scale(scaling).subtract(totalPos);
and then vector to target like
var targetVector = target.subtract(armPos).subtract(totalPos);
targetVector.rotateByQuaternionToRef(worldQuatInv,targetVector);

At this point target vector seems wrong. And really, it depends on bone absolute positions, so, sigh...

#57 - 06/13/2022 09:42 PM - Josip Almasi

... and yes, this arm position is in wrong coordinate system: {X: 0.13754705781588328 Y: -0.017311424858179172 Z: 1.6750443792873426}
for anime blue - Z is 'up' for that model. For gracy lee it's {X: 0.1751229093843536 Y: 1.6291742177001083 Z: -0.03606258881607194}
as it has Y 'up'.
In the same line position in global space is subtracted, and that's where the mess starts.
Apparently Bone.getTransformNode().getAbsolutePosition() does return actual absolute position, e.g. {X: -0.1375470608472824 Y: 1.6750396490097046 Z: -0.017316075041890144} for anime blue.
But the rest of the calculus is now off :)

#58 - 06/15/2022 10:22 AM - Josip Almasi

Dang, this microsoft kid doesn't get it. It's not a bug it's a feature :)
Vander what you think of this, what's the course of action for us?
Even reverting back to babylon 4 is non-trivial.
At this point I'd gladly just fork out and revert all of his commits...

#59 - 06/15/2022 12:34 PM - Vander Dias

I **believe** that they won't make breaking changes on the skeleton code for a long time now, so maybe the safer if you want to keep up-to-date with Babylon releases then we should find a way to make it work with the way it works now.

I would like to proceed with that PG and slowly migrate the pose handling code to that. When it breaks, step back and try in a different way, until it's migrated successfully. When finished migrating to PG, then integrate to the main code.

Just my thoughts.

IMHO reverting back would lock the project to old versions, preventing the project to take advantage of new features.

#60 - 06/15/2022 01:08 PM - Josip Almasi

Exactly my thoughts - until yesterday :)
But as of today, I think we have exactly zero support with gltf characters. This guy received his ticked from his PM, has covered this one use case that Microsoft needs, and the ticket is closed. I didn't get impression that he even bothered to look at both playgrounds, and the second one gets positions all over the place, literally.
And this is trivial case of a single position! Actual IK involves two vectors converted to common coordinate system, or three in VR, as we also have to take controller orientation into account.
Al that on top of library that seems simply broken.

Alright see if you can figure it out in playground, even proving what's broken would be a victory. I just pushed babylon5-avatars branch, I think I have positions figured out, but rotations...

Right now I'm too frustrated to think clearly. But obviously we need some sort of quality control, now that XR is also broken. So I have babylon code checked out, I'll see what it takes to build and ditch their CDN.

#61 - 06/15/2022 02:00 PM - Vander Dias

At this moment, I can get anime-blue from A-Pose into T-Pose:

https://playground.babylonjs.com/#H925CH#40

What I am now trying is to calculate for every model find the rotation from its rest position to T-Pose, so I know they're all in a known pose before I start rotating to reach user's pose.

After I succeed at that, I plan to then migrate reachFor() to this, and find a way to mimic user's pose using only relative rotations. I believe that has a chance of working.

#62 - 06/15/2022 02:18 PM - Josip Almasi

Way to go!
Funny though, in firefox it remains in A pose :)

#63 - 06/15/2022 05:32 PM - Vander Dias

I **finally** got the absolute positions:

https://playground.babylonjs.com/#H925CH#52 <<--- almost there!

Look at the console.log, positions are now different. Got it this way:

```
left_arm.getTransformNode().computeWorldMatrix(true);
left_arm.computeAbsoluteTransforms();
```

Which is, when calling computeWorldMatrix(), use the transform node, but when computeAbsoluteTransforms(), use the bone itself.

Working on it...

#64 - 06/16/2022 10:11 PM - Vander Dias

So, now that you went back to 4.x, should we continue on this task?

Still can't open the 5.x version in XR here. Trying to figure out what's wrong, but no luck yet.

#65 - 06/17/2022 01:16 AM - Josip Almasi

Why ask me, you're the expert ;)
That's to say, I trust your judgement. Here's how it looks so far to me:
Babylon 5 is simply unstable, and we can't estimate when how long will that last. On the other hand, it does bring improvements, most important being WebGPU. So we have to stay tuned, and we will have to upgrade eventually.

So, do you want to continue working on this? I.e. how does it look to you so far, waste of time or it can be done in reasonable time?
Let's catch up some time tomorrow and discuss?

Don't bother with XR in 5.10, we had it working with 5.9, babylon upgrade broke it. That was one thing too much that made me revert back.

#66 - 06/17/2022 02:37 AM - Vander Dias

Yes, let's discuss tomorrow then, that will be nice. =)

#67 - 06/28/2022 01:36 AM - Vander Dias

- Status changed from New to In Progress

#68 - 07/09/2022 02:55 PM - Josip Almasi

- Assignee deleted (Vander Dias)

#69 - 08/26/2022 05:02 PM - Josip Almasi

- Description updated

#70 - 10/22/2023 12:46 PM - Josip Almasi

- Assignee set to Josip Almasi

- Status changed from In Progress to Resolved

Solved in babylon5-avatars branch. Changes seem to be backwards compatible, i.e. work with babylon 4.
Bones do not return any relevant positions or rotations any longer, so everything had to be reworked to use bone.getTransformNode() instead. And when modified directly like with IK, that returns relevant data only after transformation matrix is recalculated. Performance impact expected, to be measured.

#71 - 10/23/2023 10:24 AM - Josip Almasi

- Status changed from Resolved to In Progress

#72 - 10/30/2023 02:10 PM - Josip Almasi

- Status changed from In Progress to Resolved

... and avatars are finally back where they were a year and half ago.
Still in babylon5-avatars branch, though not tested with babylon5 at all.

#73 - 11/11/2023 01:10 PM - Josip Almasi

- Status changed from Resolved to Closed

Files

anime-blue.zip	2.29 MB	06/09/2022	Vander Dias
avatar.js	57.3 KB	06/09/2022	Vander Dias