BlocksmithXR Essentials for advanced self-learners

Builder version 6.1.0

This document captures notes that **we want advanced users to know**, so they can have a great experience.
This is not a tutorial. Check out blocksmithxr.com for tutorials. If you are using the Education Edition, find a collection of animated lessons that start with the core aspects of the software and walk you all the way to making your own Multiplayer games. They will be assigned to you as a Quest in the Dashboard.

This is a collection of tips & tricks for **advanced self-taught users**, which should help them avoid known roadblocks and discover some of the hidden gems of the system.

For now, these tips will make your life easier and your games & experiences more lively!

Contact us at create@blocksmithxr.com with questions & suggestions!

Content

BlocksmithXR Builder Cheat Sheet

Mouse Controls:

left click = select object

scroll wheel = zoom in/out

right click + drag = orbit view

middle mouse button + drag = pan view

click and drag on object = move object

click and drag = select multiple objects

Useful Hotkeys:

Fly mode: wasd = move

Fly mode: alt + mouse = look around

spacebar = snap mode*

r = rapidly duplicate objects

f = focus on selected object

g = group or ungroup

shift + scale = proportional scale

shift + drag object = auto-stack object

VR/AR Development in 5 easy steps:

1. Build a scene using basic shapes, built-in objects, imported media or 3D models
2. Animate parts of the scene
3. Add Events to objects to make them interactive
4. Save the experience to the cloud
5. View in the Builder or on your favorite device

* Snap mode:

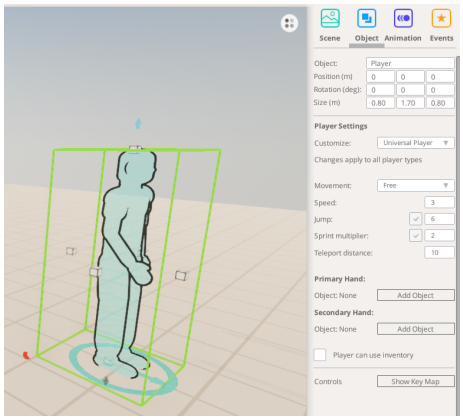
- Select snap points and drag them towards other objects to connect them.
- Drag the object's frame towards other objects. When ghost appears, release to snap object into that position.
- Use white handles to scale-snap objects, a great feature to connect walls, etc

Basic Editing

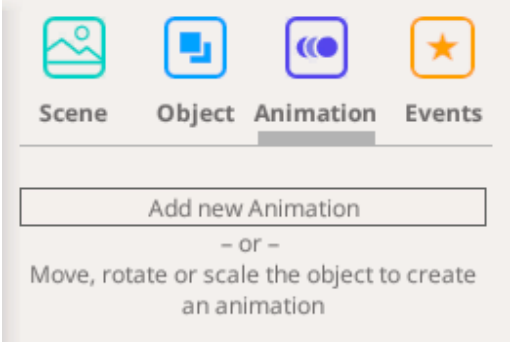
1. The **player figure in blue** that you see in an empty scene. This is important because it defines where players start and in which direction they look when a scene starts. It also helps with identifying scale.

   In AR, it defines the relative starting position/scale of the experience to the player.

   You can check your perspective (and test your game) by hitting “Play” - “Play in Viewer” and we recommend doing this early & often.



2. **Animating objects**: Select an object, click the “Animation” tab, and then drag the selected object to make the animation hologram. This defines the target position, rotation and scale.



3. **Groups** are a powerful feature to keep scenes organized. Group Editing blends out the scene to avoid clicking on objects in the background when editing and provides easy access to groups within groups
   - Use **group editing** to add an element to a group
   - Groups can be saved as models
   - Use groups for complex chained events: Imagine a player holding a gun which fires a bullet which should explode when it hits a wall. To work on that impact effect, you can select the player, click “Edit Object” in the primary hand to open the group editor for the gun, find the spawner, click “Edit Object” again to edit the bullet.

4. **Stacked Animations:**
   - To make a drone flying on a path through several waypoints, create a sequence of animations using Animation Steps

- To make a model that carries multiple independent animations (jump, crash, taunt), create several animation sequences by adding a new Animation. Those are meant to be played independently, through the event system. Only one Animation can be played at a time.

5. When working with multiple scenes, **pay attention to the scene transitions**. Per default, scenes don't switch automatically. You either need to setup an event that triggers a transition or change the transition in the scene settings to AutoPlay.

Making experiences

1. You can **change the way players can move around in scenes** by clicking on the player figure and changing movement style in the Object tab.
   Default is Free, which enables FPS controls on desktop computers (WASD keys to move and mouse to look), joystick controls on mobile devices and free teleporting with VR controllers.
   "Teleport" limits movement to teleport points (in the "Special" tab).
   "None" allows players to turn around only. Can be used for lobbies or storytelling scenes
   "Freeze" means the player can't do anything. Typically used for UI.
   All these modes can be tested with "Play" - "Play in Viewer" immediately.
   For roomscale VR and AR, players can obviously move around freely. They interact with the experience through objects with selection events and trigger their slots with gestures. Note that any limitations on the Player won't work in VR.
   Also note that "Freeze" means the camera is basically attached to the player face in VR. Do not do this for VR experiences!

2. **Player figures have slots**. These are mapped to the typical buttons on the various controllers and can be used to enable jumping or speed-ups and they also can hold objects with "Interaction" events.
   Example: Build a small sci-fi gun with some shapes and a Spawner (from the special object tab). Make it spawn an object with a "On Interaction start" event. Place it next to the player.
   Add the object to "Primary Hand" or "Secondary Hand". It will jump into the players left or right hand and in play mode will fire an object every time the button is pressed.
   The head slot can be used for anything that the player carries around attached to the head/body movement.

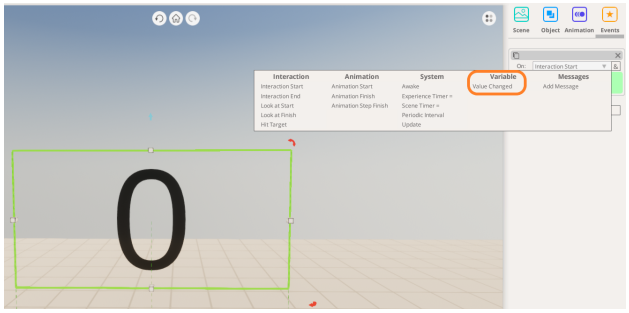
3. Sounds add a lot to experiences but are often an afterthought. **Use "Add component" to add sounds to objects**.
   Use Events to play sounds at specific times, or use the Play Sound at Start checkbox in the Object tab to make them play immediately when a scene starts.

4. All sounds in BlocksmithXR are spatial (3D). If you need a 2D ambient sound, use the scene sound setting in the “Scene” tab. Alternatively, add a Microphone object to a scene and enable “Use Microphone” on the Player. The microphone will broadcast local sounds to the players.

Events/Logic

1. Many objects have **special events and actions**.
   Example: Variables have a “On Value changed” event and a “Set to” action, among others.

   Other cases: Spawners have a “Spawn Object” Action. Trigger objects have enter/exit events.



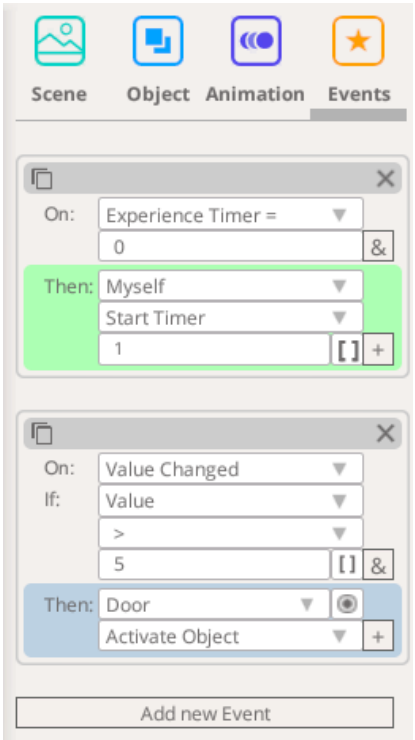
2. Events are executed top to bottom

3. Events can be combined with further conditions by using the “&” button. Those can have **special conditions for some events**. For example, the “Hit Target” event has a bunch of conditions that can further define what got hit.

4. **Timers are extremely powerful for games**. Use Variables from the “Special” tab or Components for that:

   This is how to setup a timer that starts ticking when the scene/game starts.

   This segment sends an “activation” message to the door object after the timer is at 5s. The door object can react to that with the “Awake” event.



5. **Variables show their values in Play Mode but can be hidden** after everything works as expected

6. **Timers have a timestep variable**. If you want to make a spawner with a cooldown of 0.3s, you can use a variable with a timestep of 10 and check for the value to be 3.

7. **Variables can be connected to objects and act as checkpoints**, forwarding activation-events only when certain conditions are met. You can setup powerful puzzles and mechanisms with that.
   Example: Animate a door to open. Add a button next to the door. When selected, make it trigger the door animation. Straightforward. But there is more.
   Add a variable and set it up as a timer. And instead of activating the door opening animation when the red button is selected, make it activate the variable.
   On the variable, add an activation event that checks the value and only if it is bigger than 5, start the door opening animation. Et voila, you have a timed door.

8. **Variables can be used to work with states**, in step by step scenarios we recommend using a variable to act as a state controller, sending a custom message to all actors in the scene whenever a state change happens. Actors are then responsible for reacting to new states
   Example: You want to create an interactive training with a conversation going like this:
   1. *Character walks up to other character*
   2. *Then starts talking*
   3. *When finished, the other character responds*

   This can be split into three states. At the beginning, the state variable is set to 1 (real programmers obviously start with 0) and sends a "StateChanged" custom message to both characters. One character receives it and starts walking, the other character receives it and does nothing. After the walking character has arrived, it sends a "ChangeState" message back to the state variable which reacts to it by increasing the state value and sending that information out to all actors again.
   This concept can be used to create breaks (using timers) and story branching
9. **The Update loop**: An Update loop is a cycle built into the BlocksmithXR engine. It generally repeats 60 times per second, on most computers. However, it can run at different speeds on different devices. For example, the HTC Vive runs an update 90 times a second, instead of 60. Keep in mind that it can run at different speeds on different devices when using the Update Event!

Collisions

1. **HitTarget event**: You can test if a moving object (bullet, arrow, magic fireball, etc) hit something, like a player or a wall. Use the “HitTarget” event on those moving objects to run events after it collides with something else. HitTarget will not fire if used on static objects.

2. **Triggers**: You can detect if an object or Player enters a certain area by using a Trigger from the “Special” tab. Triggers immediately detect when an object enters or exits them, and then can apply certain events.

3. **Collidable** Advanced Option: Most objects are solid (aka. collideable) by default. But any object can become un-collidable by disabling a checkbox its Advanced Options. This will allow the users to move through those objects, while they are still visible.
   Note that this option only affects the physical “can I bump into this” collider. Triggers will trigger independently if this option is on or off.

4. **Collide as Group** setting: Any group has the option to collide with other objects in its entirety, rather than as its component objects. This is highly useful in some cases. For example, if a soccer ball made from five shapes enters a goal, you would want to enable Collide as Group so only one point is scored, instead of five!

5. **Collidable objects can push the Player in VR**: This setting allows for two different Player-collision behaviors in VR. In some cases, you want moving objects to push the player (elevators, moving walls, etc). But in other cases you absolutely do not want collidable objects to push back on the player (eg. architecture tours). Toggle between those two behaviors to suit your needs.

Multiplayer

1. **Multiplayer lobbies** (from the “Special Objects” tab) expect a scene after the scene they got placed in. When configured, there will be several player figures in the next scene, representing each team and each player type. They define spawn points and loadout for the various team members
   Setup: In the lobby scene, create a “On Selection Start” Event on a button or textbox, choose “Other object”, connect it with the multiplayer lobby and choose “Find public game”

2. **Play in Viewer** detects multiplayer setups and offer to test from the various players’ perspective.

3. Multiplayer games can be played together with the Play in Viewer functionality, so several users can download a public experience from another user into their Builder software and play it from there. It’s more fun on dedicated devices, but in fairness, a desktop PC is a great device for multiplayer games right there, so playing right from the Builder was an important feature for us.

4. **Multiplayer games between VR & non-VR players.** VR players have different properties than non-VR players. They have a much better overview of a scene. Movement is often limited though. Teleporting however is incredibly strong (try to shoot someone who is teleporting).
   Try to design your games utilizing those different powers instead of artificially restricting them. For example, limiting a VR players teleport ability with a timer in a multiplayer paintball game might sound natural, but it’s not really fun.

5. **Multiplayer games between AR & non-AR players.** AR players again have different properties than non-AR or VR players. Don’t try to immerse them into a full size environment, rather use a table-scale display of whole scenes or individual animated objects to depict game mechanics and interactions

6. **Multiplayer games between AR & VR players.**
   We have seen some fascinating approaches and game design concepts there. Make sure you share your experiments with us!

Tips & Tricks for VR/AR development

1. **Test early**. A 1x1x1m cube can look small on the screen but can appear larger when you are standing next to it. In VR, users will feel as if THEY ARE in your experience, making them comfortable there is important. Our path from creating to viewing on devices is very fast/smooth and we encourage using this for early testing.
   Similarly, most AR experiences are being resized to table scale when played. Find the right scale level early and decide if your players should be able to resize to their liking.

2. **Player size matters**. The default player in the Builder will map the actual player height in VR 1:1. A tall human player in VR will look at the scene from an elevated, a short human player from a lower point of view. Changing the default player in the Builder will make the real player feel proportionally taller or smaller in VR. This is great if you want to make a giant out of the player. However for architecture and model VR reviews we strongly recommend to not change the player size and to do precise floor calibration, otherwise your clients will feel like they have the wrong height when viewing the experience.

3. **Player rotation is important**. The moment they spawn in your experience, they will start to look around. For scene changes and when teleporting you need to decide if you enforce a rotation change for players or preserve where they are.
   Example: You make an elaborate horror experience with a monster sneaking up on the players, but when they spawn in your scene they are accidentally facing the direction of your monster => effect ruined
   Tips: Use visual cues to guide players. Use the “looked at” event to identify where your players are.

EDU Instructor Notes

In our test classes we found that offering students a very easy way to put a large amount of objects into a 3D scene, they will, surprise, surprise, put large amounts of objects into a 3D scene. Our Model Pool is an especially tempting candidate there.
This will cause slow computers to bog down and their creations won't work well in VR/AR, causing frame drops aka stuttering which is known to cause motion sickness. We are going to enforce more discipline with the help of the performance meters and warning popups but recommend framing this early in the teaching process:
VR/AR is ALL about performance.

Typical classes start out with some level of 3D modelling/construction. When going into games, students often don't make the connection to their early experiences. They can go back though into what they created earlier and save their models locally (or share them) and re-use them in their games, which we found they really liked.

A student creating a multiplayer game, then having other students, even teachers, join and play against each other creates a level of fun & excitement that is rarely seen in classrooms and camps.
We wanted to share an observation we've not been expecting:
Playing a simple multiplayer battle in the same room with others is exhilarating. This is not kid level fun. It's exciting for kids & grown-ups alike and creates funny situations, laughter and bonding, all in a matter of minutes.
We found that this was incredibly rewarding for the student whose game got played (usually the one who finishes first).
But also, every student did not get the chance to have their games played. Some had not finished and went into frantic mode to get there (which just amplified mistakes and was very frustrating for them).
We recommend making enough time for the joint playtesting phase and pacing it so that advanced kids can polish their games while everybody gets a chance to get their games played.