Recent Updates Corrected flexbody issue Jan 18, 2024 V3.6 Animated Skins Transform your vehicles into molten machines with the Universal LAVA skin! This fiery texture gives your car a red-hot, lava-like appearance, making it stand out on any track. Whether you're cruising or crashing, this skin adds a sizzling touch of style and intensity to your BeamNG Drive experience. Embrace the heat and set the road on fire! To make the glow work add the Paint Design "Ashmakers Lava Glow" to your vehicle -> Go to the Colour tab and then choose your base colour for Colour1, then Colour 2 and 3 set to a lighter colour to make it glow. Animated Skins for ALL vehicles and some trailers: You can now be transparent but with the lava glow: Credit: @nobeam - For creating the emission layer and helping. @Shadows - For helping with making it colourable. @stefan750 - For help with the lua code @Butch13 Gaming - For helping with the layers Any problems/issues/suggestions head over to my Discord server. DO NOT RE-UPLOAD OR UPLOAD TO ANOTHER SITE Glow maps (glowMaps) allow you to switch between multiple materials based on a number value. The most common use case of glow maps is vehicle lights, hence the name. This includes headlights, taillights, gauge lights, etc. In those cases they are used for changing the light materials to emissive variants of themselves. Check this page for more information about light materials. However, they can be used for any kind of application that involves changing a material on the fly. Syntax Unlike most other Jbeam sections, the glowMap section is not a table but rather a dictionary. The key is the name of the associated material as set in the modelling software, while the value is another dictionary. It can contain multiple different dictionaries inside of it which will be described later, along with 3 values referred to as “material slots”, under the keys “off”, “on” and “on_intense”. Their values are different materials the part will be using based on the resulting value of simpleFunction or advancedFunction. "glowMap":{ "taillights":{ "simpleFunction":{"lowhighbeam":0.49,"brake":0.49}, "off":"lights", "on":"lights_on", "on_intense":"lights_on_intense" }, } While glowMaps support applying a material with the same name as their own key, it is recommended to avoid this, to prevent possible unwanted behaviour due to changes within the game. The material defined as the glowMap key does not need to be added to the vehicle’s materials.json files as it is generated automatically for each vehicle - however, for materials present in the “common” folder it is still necessary as they are not assigned to a single vehicle but shared between all of them. For this purpose, dummy materials are used. Available material slots Each glow map offers the possibility for 3 different switchable materials, referred to as “off”, “on” and “on_intense”. The switch between modes works by adding together a few inputs using multipliers. If the total value is less than 0.0001, the glowMap will be “off”, between 0.0001 and 0.5 will be “on”, and above will be “on_intense”. The function result can be greater than 1. Requirement value is 0.0001 or lower Requirement value is above 0.0001 and below 0.5 Requirement value is 0.5 or larger “on_intense” is only cared about when it is specified. Otherwise, “on” will be used. Glowmap Value Calculation A typical vehicle glowMap section looks like this: "glowMap":{ //exterior lights "signal_L": {"simpleFunction":{"signal_L":0.49}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, "signal_R": {"simpleFunction":{"signal_R":0.49}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, "taillight":{"simpleFunction":{"lowhighbeam":0.49,"brake":1},"off":"light","on":"light_on","on_intense":"light_on_intense"}, "chmsl": {"simpleFunction":{"brake":100}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, "foglight": {"simpleFunction":{"fog":1}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, "headlight":{"simpleFunction":{"lowbeam":0.49,"highbeam":1}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, "reverse": {"simpleFunction":{"reverse":0.49}, "off":"light","on":"light_on","on_intense":"light_on_intense"}, //gauge faces and needles "gauges": {"simpleFunction":{"lowhighbeam":0.6}, "off":"gauges","on":"gauges_on"}, //gauge lights "signal_L": {"simpleFunction":"signal_L", "off":"invis", "on":"decals_gauges"}, "signal_R": {"simpleFunction":"signal_R", "off":"invis", "on":"decals_gauges"}, "checkengine": {"simpleFunction":"checkengine", "off":"invis", "on":"decals_gauges"}, "hazard": {"simpleFunction":"hazard", "off":"invis", "on":"decals_gauges"}, "battery": {"simpleFunction":"battery", "off":"invis", "on":"decals_gauges"}, "highbeam": {"simpleFunction":"highbeam", "off":"invis", "on":"decals_gauges"}, "parkingbrake":{"simpleFunction":"parkingbrake", "off":"invis", "on":"decals_gauges"}, "lowfuel": {"simpleFunction":"lowfuel", "off":"invis", "on":"decals_gauges"}, "abs": {"simpleFunction":"abs", "off":"invis", "on":"decals_gauges"}, } The entire glowMap functionality depends on Vehicle Lua electrics values. See here for a list of available defaults. It should be noted that while most of the values will always be 0 or 1, some of them can return inbetween numbers. For example “brake” is equal to the braking input, meaning the value can be anywhere between 0 and 1. Keep this in mind when defining the weight of the various functions to make sure they work as expected. The value can be calculated in 3 different ways based on those values: Directly taken from a single electrics value, via the binary variant of simpleFunction Summed up from multiple electrics values with multipliers, via the complex variant of simpleFunction Calculated from a Lua code snipped, provided inside advancedFunction Binary simpleFunction If the “simpleFunction” argument is provided and is a string, it is expected to be the name of an electrics value. This sets the glowmap value to exactly this electrics value without any modifications or calculations. For example, this glowMap will use the “off” material when the brake is not in use. As soon as the brake is enabled, the “on” state is used. "glowMap":{ "brakelights":{"simpleFunction":"brake", "off":"lights", "on":"lights_on"}, } Complex simpleFunction The “simpleFunction” argument may also be a dictionary containing the names of electrics values, each assigned to a multiplier they add to the glowmap value. Negative modifiers are supported and the total result can be greater than 1. This is the most common method of calculating the glowmap value. For example, this glowmap will use the “off” material when the lights are off and the brake is not in use. As soon as either are enabled, the “on” state is used. When both are enabled, the “on_intense” state is used. If this state is not defined, the “on” state will replace it. "glowMap":{ "taillights":{ "simpleFunction":{"lowhighbeam":0.49,"brake":0.49}, "off":"lights", "on":"lights_on", "on_intense":"lights_on_intense" }, } It should be noted in that specific case that while lowhighbeam will always be 0 or 1, brake is equal to the braking input, meaning the value can be anywhere between 0 and 1. Advanced Example This simpleFunction creates a combined DRL/signal glowmap value. When the engine is enabled, “running” is set to true. This adds 0.49 to the glowmap and switches the state to “on”. When the right signal is activated, “signal_right_input” is set to true and 1 is subtracted from the glowmap value, making it use the “off” material despite “running” being true. “signal_R” is activated and deactivated based on a timer to achieve a blink effect. When it is active, it adds 2 to the glowMap value, resulting in a value higher than 0.5. "glowMap":{ "running_signal_R":{ "simpleFunction": { "signal_right_input": -1, "running": 0.49, "signal_R": 2 }, "off":"lights", "on":"lights_on", "on_intense":"lights_on_intense" }, } advancedFunction advancedFunction supports using a straight Lua code snippet to calculate its value. This gives you direct control over the glowmap without requiring an external Lua module. This feature is not often used, the simplest use case is for a gear display. This method requires a dictionary with the following keys: The table syntax can be used in case of a single value: The dictionary syntax is used with multiple values. Each of them needs a unique key per advancedFunction section, however the names of the keys don’t matter as they are disregarded in processing. "triggers":{"1":"gear","2":"gearIndex"}, With a single value you can also omit the key by setting it to an empty string: For example, this advancedFunction switches to “on” when the park gear is selected. "glowMap":{ "citybus_transmission_DSP_P": { "advancedFunction":{ "triggers":{"":"gear"}, "cmd":"gear=='P' and 1 or 0" }, "off":"invis""on":"citybus_transmission_DSP" }, } And this one switches to “on” when the transmission is in 2nd gear. "glowMap":{ "citybus_transmission_DSP_2": { "advancedFunction":{ "triggers":{"":"gearIndex"}, "cmd":"gearIndex==2 and 1 or 0" }, "off":"invis", "on":"citybus_transmission_DSP", }, } Advanced Example This advancedFunction emulates the behaviour of the combined DRL/signal simpleFunction showcased in the advanced example above. "glowMap":{ "running_signal_R":{ "advancedFunction":{ "triggers": { "1": "signal_right_input", "2": "engineRunning", "3": "signal_R" }, "cmd": "engineRunning*0.49 + signal_R*2 - signal_right_input" } } } Material transitions For the purpose of realistic halogen and sealed beam lights, it is possible to define transitions between the states of the lights. This works by modifying the emissiveFactor of the glowing light materials via a smoother . In the glowMap section, this is achieved by using special electrics values defined in the electrics section, and specifying its threshold value where the emissiveFactor of the “on” material is at its maximum value, which is done in the “materialEmissiveScaling” argument. "glowMap":{ "sealedbeam_auxlight": { "simpleFunction":{ "foglight_filament":0.49 }, "off":"sealedbeam", "on":"sealedbeam_on", "materialEmissiveScaling":{"on_max":0.49} }, "sealedbeam_auxlightglass": { "simpleFunction":{ "foglight_filament":0.49 }, "off":"sealedbeam_glass", "on":"sealedbeam_glass_on", "materialEmissiveScaling": {"on_max":0.49} }, } Multiple input values are also possible the same way as they are used with standard electrics functions. Thanks to the dynamic emissiveFactor of the material, the “on_intense” state of the lights is not needed here - “on” will be used as the maximum value, and the middle state will be the transition between “off” and “on”. "glowMap":{ "sealedbeam_headlight": { "simpleFunction":{ "lowbeam_filament":0.49,"highbeam_filament":1 }, "off":"sealedbeam", "on":"sealedbeam_on_intense", "materialEmissiveScaling":{"on_max":1} }, "sealedbeam_headlightglass": { "simpleFunction":{ "lowbeam_filament":0.49,"highbeam_filament":1 }, "off":"sealedbeam_glass", "on":"sealedbeam_glass_on_intense", "materialEmissiveScaling":{"on_max":1} }, } Last modified: January 24, 2025 As you may notice, the code for this is borrowed from another mod of mine, the Gavril Vertex How it works: The brakes have a new overlay mesh added around them which has a transparent HTML texture. The texture will start to change the transparency of 3 texture layers once certain temperatures have been reached. As it's an HTML screen, this can be done very smoothly as shown in the GIF. Thanks to LucasBE , the textures now look A LOT better as they're properly rendered and stuff I adjusted the temps a bit downward to make the brake glow more noticable, so it's not that rare occasion that you can sometimes see when looking at it in the right moment. However, not that low that its always there. Heavy braking may lead to a short moment of red glow. Proceeding to do so will obviously heat-soak the brakes etc. Currently Supported base-game vehicles: Bruckell LeGran Bruckell Moonhawk Bruckell Bastion Burnside Special Civetta Bolide Civetta Scintilla Cherrier Vivace/Tograc ETK I-Series ETK K-Series ETK 800-Series Hirochi SBR4 Hirochi Sunburst Gavril D-Series Gavril Roamer Gavril H-Series Gavril Barstow Gavril Bluebuck Gavril Grand-Marshal Ibishu Pigeon Ibishu Wigeon Ibishu Miramar Ibishu Pessima (old) Ibishu Pessima (new) Ibishu Hopper Ibishu Covet Ibishu 200BX Solaid Wendover SP Dunekicker SP Rockbasher All base game vehicles (apart from the Piccolina) that have Disc Brakes are supported in this mod. Picoolina isn't supported as that still uses it's own ancient and outdated brake meshes and textures instead of the common ones Credits: @aljowen for the base of the HTML that I could work on @13Stewartc for being kind and explaining how the HTML worked for it's original intend @LucasBE for the new glow textures Anyways, that's all there is to this little mod, have fun! Feel free to leave a review If you want to support me a bit financially, you can leave a donation if you want. Also check out my other mods If you have any issues, please MESSAGE me or write into the Discussion Thread of this mod! PLEASE KEEP IN MIND THAT YOU HAVE TO APPLY THE GLOW MANUALLY IN THE REPOSIORY VERSION OF THIS MOD! IT IS NOT BROKEN Also feel free to implement the glow into your own mod vehicle(s)! Message me so I can add your mod to the supported mods list in the thread about this mod and explain how to properly implement it. Don't just take this and put it in some mod without asking, especially if it's a paid one Mods > Skins > Dismiss Notice In the upcoming weeks mod approvals could be slower than normal. Thank you for your patience. Tags: Overview Updates (2) Version History New custom Skin design for cars and trailers! Under paint design look for (Glow) Any problems/issues/suggestions head over to my Discord server. DO NOT RE-UPLOAD OR UPLOAD TO ANOTHER SITE Mods > Mods of Mods > Dismiss Notice In the upcoming weeks mod approvals could be slower than normal. Thank you for your patience. Overview Updates (2) Reviews (11) Version History