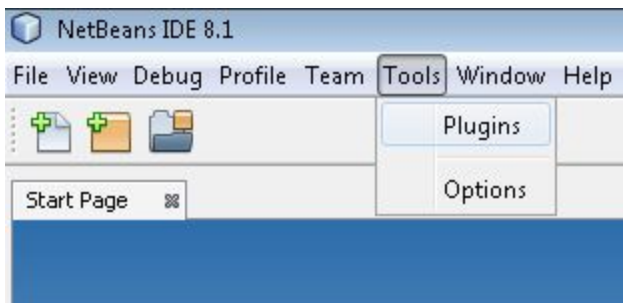
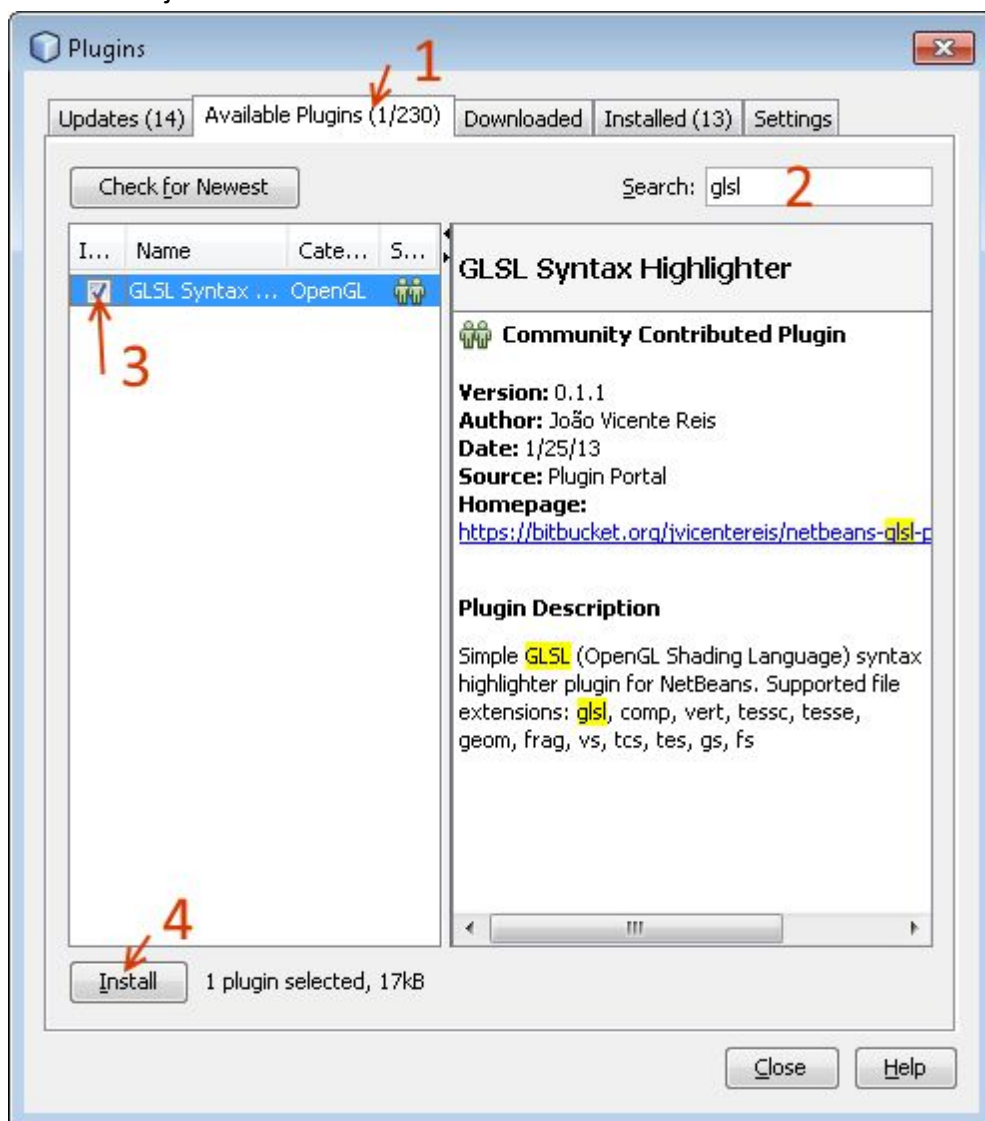


Příprava NetBeans

Pro příjemnější práci se shadery si nainstalujte “GLSL Syntax Highlighter” plugin. V horním menu Tools → Plugins.

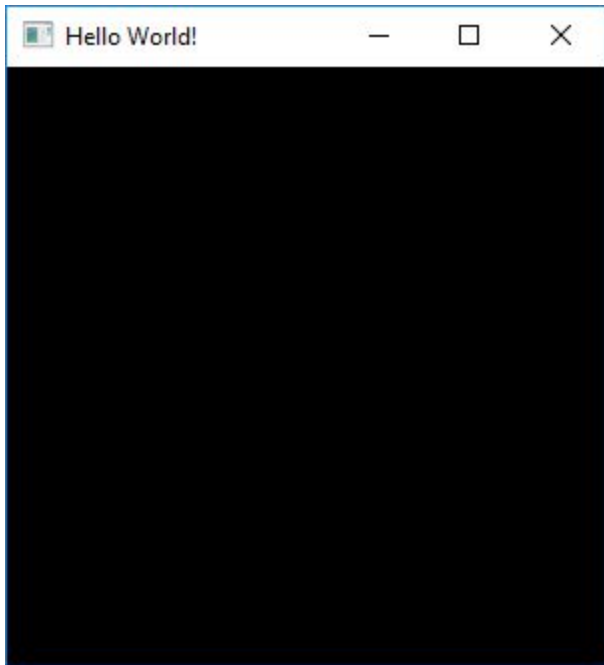


Pak na záložce “Available plugins” vyhledejte “glsl”, zaškrtněte “GLSL Syntax Highlighter” a klikněte na “Install”. Dále už je to klasický postup “next, next, accept, continue apod.”. Nakonec se restartuje NetBeans a je hotovo.



PV112 Java - 1. cvičení

Stáhněte si cv.zip ze studijních materiálů. Rozbalte a složku otevřete v NetBeans File -> Open project... (Ctrl + Shift + O) . Po úspěšném otevření, zkompilování a spuštění projektu (Ctrl + F5) by se mělo zobrazit okno 300x300 pixelů uprostřed obrazovky s nadpisem "Hello World!" a černým pozadím.



Dnes budeme pracovat převážně v souboru Cv1.java, který v NetBeans najdete v projektovém okně pod cv1 → Source packages → cv1 → Cv1.java .

Po dostatečném zorientování ve struktuře třídy doporučuji stiskem Ctrl + Shift + Minus sbalit všechny bloky a rozbalit si pouze metody `init()` a `render()` .

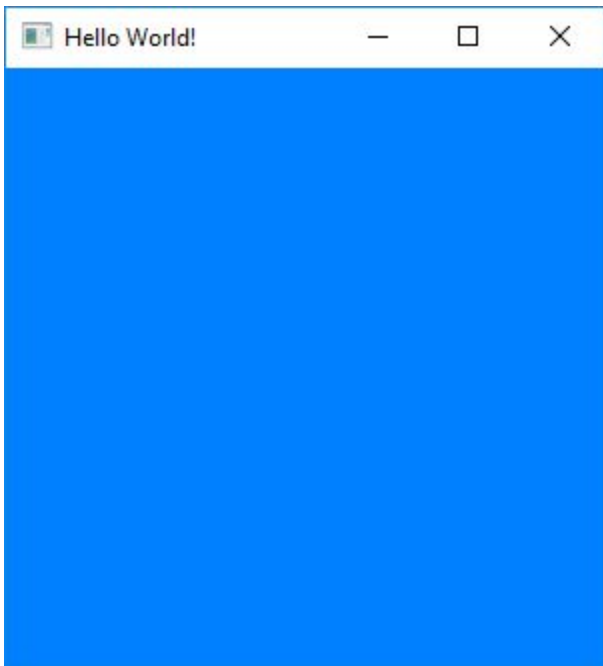
Task 1 - barva pozadí

Barva pozadí vzniká tak, že se framebuffer (zobrazovací paměť okna) nastaví / vynuluje nějakou barvou. To se děje voláním `glClearColor`, které je zatím jediným příkazem ve funkci `render`.

Barva, kterou se paměť přepíše se ale nastavuje pomocí `glClearColor`, jelikož barvu pozadí nehodláme během vykreslování měnit, umístili jsme ho do funkce `init()`, která se volá jen jednou na začátku.

Zkuste si pohrát s parametry `glClearColor` a nastavit si tak pozadí okna na svoji oblíbenou barvu.

Příklad: `glClearColor(0.0f, 0.5f, 1.0f, 1.0f);`



Task 2 - geometrie

Vykreslíme si svůj první objekt - trojúhelník.

Pro to musíme vytvořit `GL_ARRAY_BUFFER` se souřadnicemi jednotlivých bodů (jsou připravené v poli `VERTICES`). Ten bude částí `VertexArray` objektu, který může definovat i další vlastnosti vykreslované geometrie (o tom více později).

Dále pak musíme si pak musíme vyžádat, aby se z tohoto zdroje dat použily 3 body k vykreslení trojúhelníku. (a ne třeba čáry nebo jednotlivých bodů).

Zajistěte vše potřebné pro vykreslení trojúhelníku, nápovědy jsou umístěny v komentářích v kódu.

Nápověda: Všechna místa v kódu, kde je potřeba něco udělat lze snadno najít vyhledáním "Task 2" v celém projektu. (Ctrl + Shift + F).



Task 3 - barva geometrie

Nyní si trojúhelník obarvíme. Zatím je bílý, protože to říká náš vertex shader. Ten najdete v `resources.shaders → vertex.glsl`. Abychom mohli měnit barvu z Java kódu, je třeba, aby vertex shader nenastavoval barvu na tvrdo pro všechny vrcholy, ale podle vstupního parametru. Pro změnu z kódu je nutné zadat data pro tento parametr. Na to stačí přidat další `GL_ARRAY_BUFFER` s daty o barvě každého bodu (pole `COLORS`). A ten odpovídajícím způsobem přidat do vykreslovaného VAO.

Zajistěte vše potřebné pro obarvení trojúhelníku, nápovědy opět v komentářích označených “Task 3”.

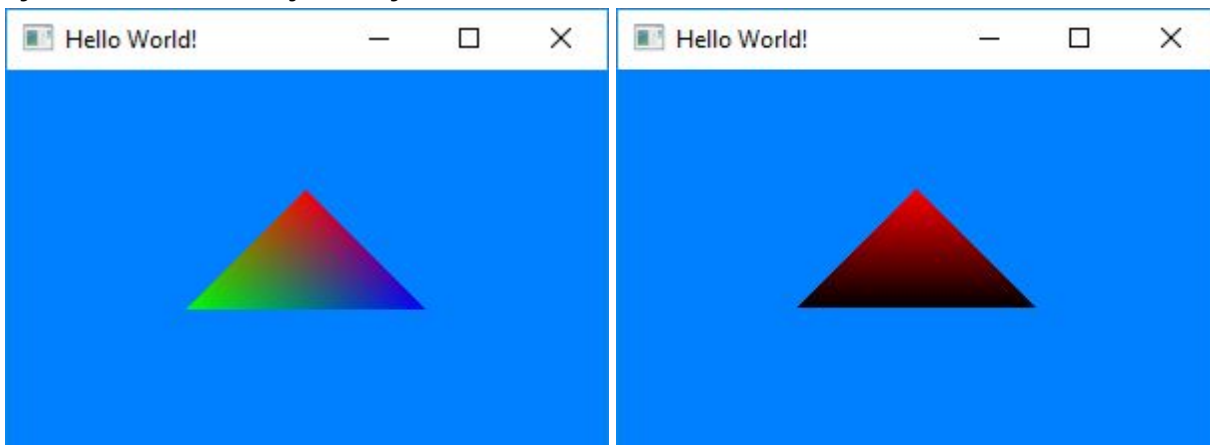
Trojúhelník by potom měl být červený.



Task 4 - interpolace barvy

Uděláme trojúhelník zajímavější tak, že nepoužijeme stejnou barvu pro všechny 3 vrcholy, ale dáme každému jinou. OpenGL totiž interpoluje hodnoty pro jednotlivé pixely trojúhelníku ze všech tří hodnot v závislosti na poloze (vzdálenosti) od vrcholů.

Zkuste si pohrát s hodnotami v poli `COLORS` tak, aby trojúhelník měnil barvu mezi vrcholy, měli byste docílit následujících výsledků.

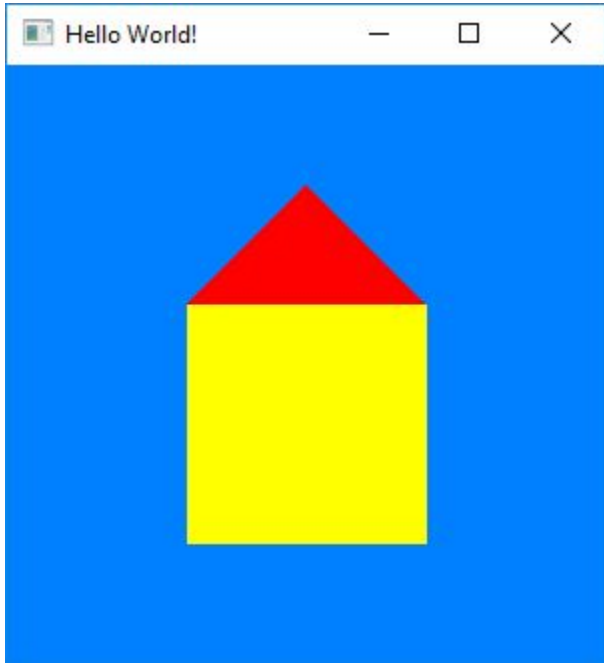


Task 5 - domeček

Pod střechu trojúhelníku přidáme budovu domečku ve tvaru čtverce. OpenGL od verze 3.1 už pro jednoduchost neumožňuje kreslit čtyřúhelníky, ale pouze trojúhelníky. Tudíž budeme potřebovat další 2 trojúhelníky.

Přidejte data pro další 2 trojúhelníky a uprave vykreslovací příkaz tak, aby se použily.

Měl by se objevit žlutý domeček.

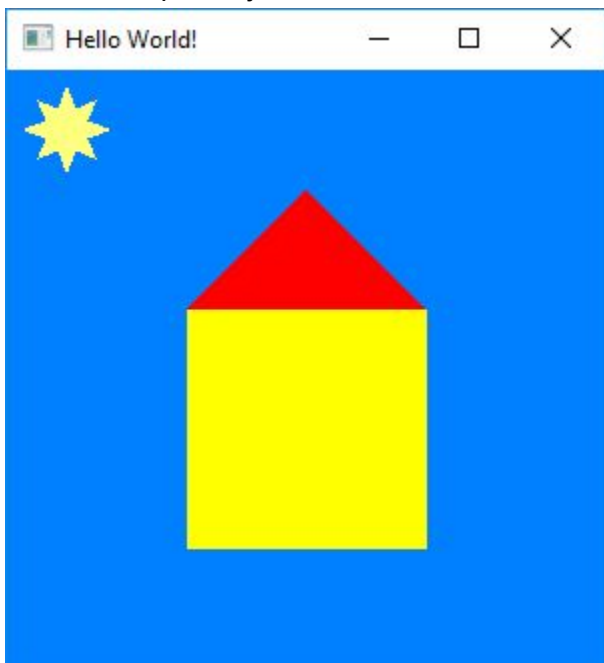


Task 6 - sluníčko

Našemu uměleckému dílu už chybí pouze nějaké líbivé pozadí. Vytvoříme dojem krásného slunečného dne vykreslením slunce na obloze. Opět ho složíme z trojúhelníků, tentokrát 16. Protože trojúhelníky budou vždy sdílet jeden bod (střed) a také jeden bod s předchozím trojúhelníkem, použijeme mód `GL_TRIANGLE_FAN`. S ním nebudeme potřebovat 16×3 bodů, ale pouze $2 + 16a$ a ušetříme tím lehce data, která posíláme na GPU.

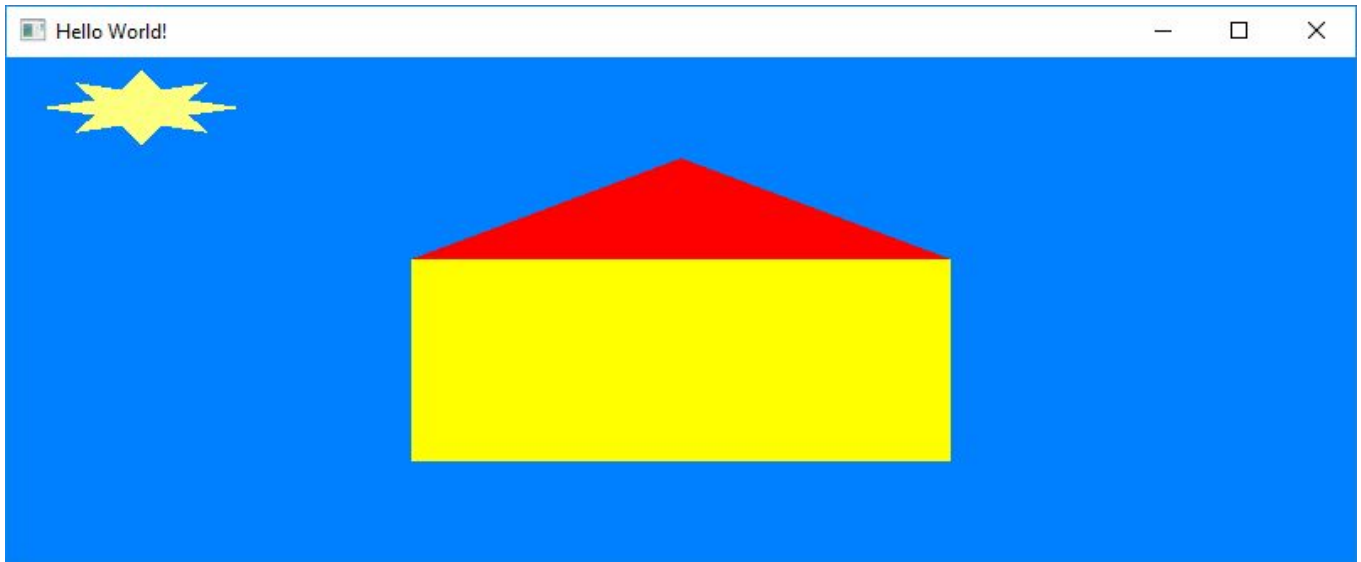
Přidejte data pro dalších 18 bodů a vykreslete je jako vějíř.

Sluníčko se pak objeví v levém horním rohu.



Task 7 - přizpůsobení poměru stran

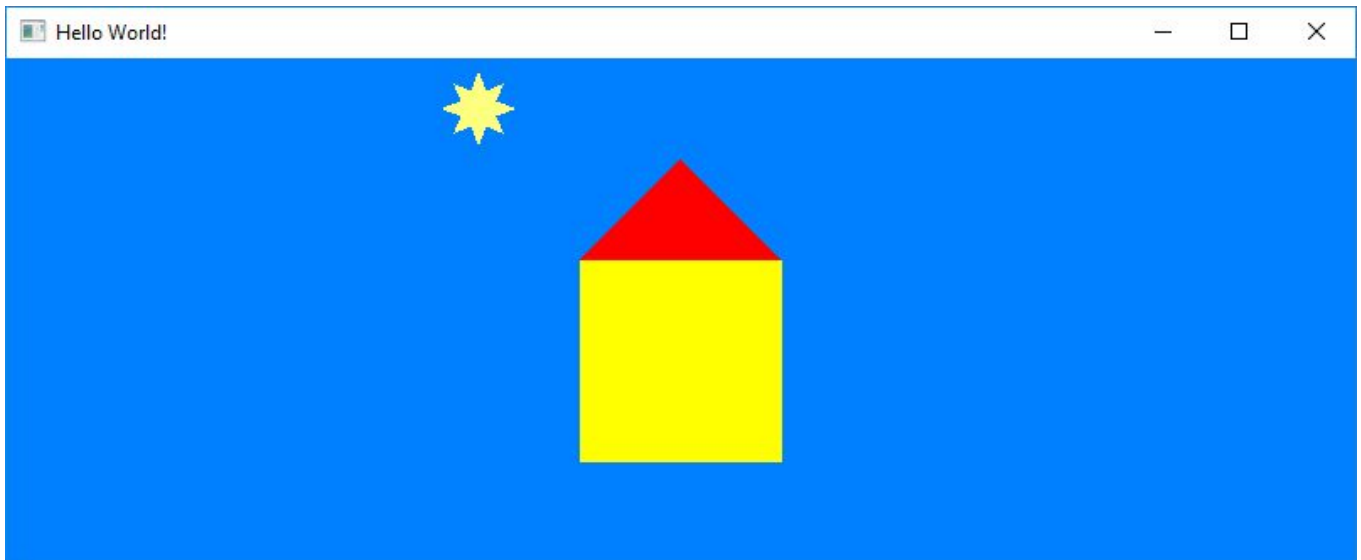
Zvídavější si již mohli všimnout, že úplně nesedí tvary objektů, pokud změňme velikost okna tak, že už to není čtverec. To je způsobeno tím, že souřadnicový systém $[-1, 1] \times [-1, 1]$ je jednoduše natažen přes celé okno. A tudíž se nám osa x roztáhne spolu s oknem.



To můžeme zvrátit jednoduchým trikem. Do vertex shaderu si předáme hodnotu poměru stran $\text{šířka}/\text{výška}$. Tou následně vydělíme výstupní x a tím osu jakoby *smrskneme* a vizuální poměr obou os bude opět totožný.

Tato hodnota se během jednoho kreslení nebude měnit. Tudíž nemá cenu ji posílat pro každý vrchol zvlášť jako vstupní parametr. Pro tyto případy jsou v OpenGL připraveny `uniform` proměnné. Tu je potřeba vytvořit v shaderu, zjistit si její OpenGL index (*location*) a na něj zapsat požadovanou hodnotu.

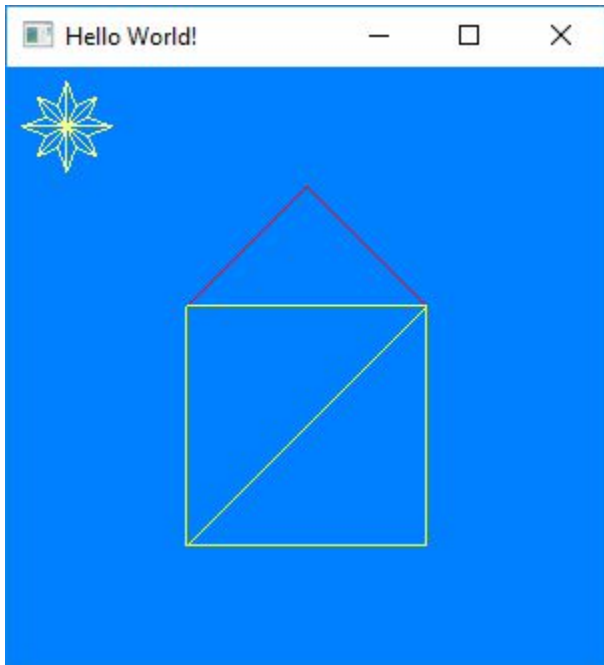
Opravte poměr stran vykreslovaných objektů pomocí `uniform` proměnné.



(bonus) Task 8 - drátěný model

Nyní se všechny trojúhelníky kreslí vyplněné, tj. jejich vnitřní plocha se vyplní pixely. To lze změnit a vykreslit pixely pouze na jejich okraji. Díky tomu uvidíte umístění jednotlivých trojúhelníků. To se může hodit později při řešení problémů při vykreslování složitějších objektů.

Zkuste před vykreslením změnit mód vykreslování polygonů (trojúhelníků) voláním `glPolygonMode`.



(bonus) Task 9 - vstup z klávesnice

Většina aplikací vyžaduje nějakou interakci s uživatelem. Často budeme reagovat na stisk klávesy nějakou změnou ve vykreslování, pro začátek zkusíme přepínat mezi drátovým a vyplněným modelem pomocí klávesy "L". Vstupy nám zprostředkovává knihovna GLFW, která nám i vytváří okno a s ním OpenGL kontext. Teď nás zajímá hlavně to, že o stisknutých klávesách nás informuje voláním anonymní funkce, kterou jsme předali funkci `glfwSetKeyCallback` v metodě `initGLFW`.

Přidejte do kódu, který bude při stisknutí klávesy L měnit hodnotu booleovské proměnné `drawWireframe`.

Podle té potom před vykreslením geometrie do snímku nastavte `glPolygonMode`.

(bonus) Task ? - slunce je hvězda

Ve světle nedávných astronomických závěrů je potřeba upravit vykreslovanou scénu, aby cípy našeho sluníčka byly ostřejší a “rozmazenější” a připomínaly tak více hvězdu na obloze .

K tomuto tasku nejsou pomocné komentáře. Stačí pouze změnit hodnoty barvy všech bodů slunce, kromě toho středového. Na jakou barvu vnější vrcholy nastavíte?

Využijte interpolaci barev pro vytvoření cool efektu.

