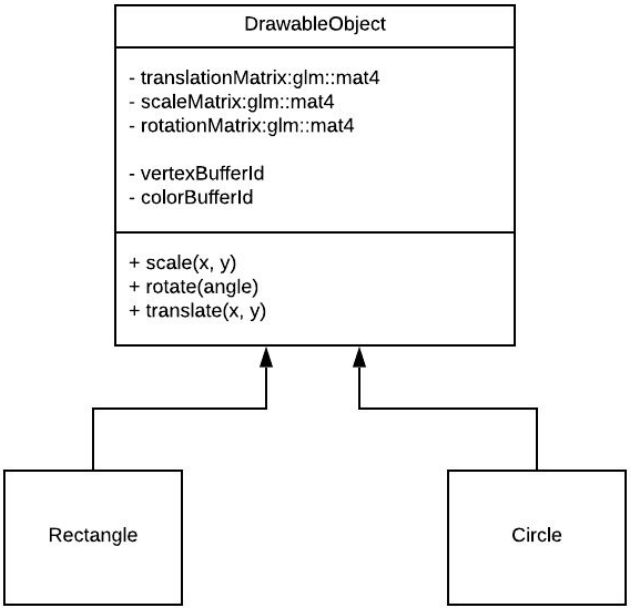
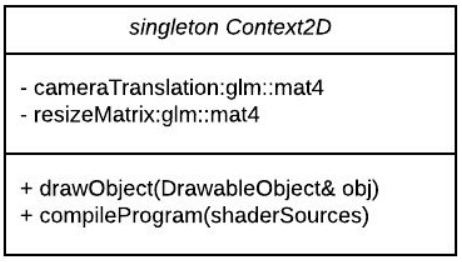
TEMA 3 - Depăşire -
Documentaţie Dinca Vlad Andrei grupa 344

Tema aleasă - Depăşire de maşini - a fost implementată în OpenGL-ul nou, în limbajul C++ şi implică transformări geometrice, atât de model (aplicate maşinilor, dreptunghiurilor, elementelor desenate) cât şi de vizualizare (translaţia camerei, resize).

Aceste transformări sunt reprezentate cu ajutorul matricelor 4x4 şi manevrate intern de către fiecare clasă ***DrawableObject*** (transformările de model, aplicate fiecărui "obiect reprezentabil" în parte) sau clasa de tip *singleton:* ***Context2D*** (ce menţine în interiorul său translaţia de cameră şi matricea de resize).



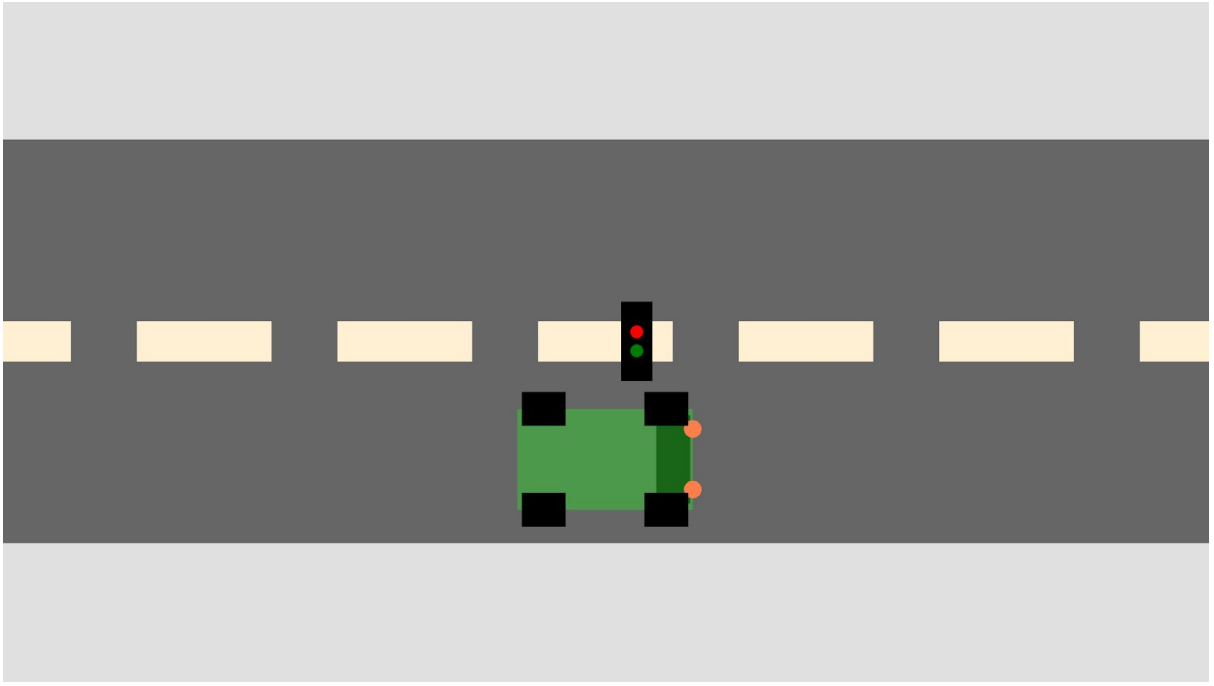
Clasa ***Context2D*** este responsabilă cu "desenarea" obiectelor de tip ***DrawableObject***. Aceasta conţine metoda ***drawObject(DrawableObject&)*** ce leagă bufferele interne ale obiectului reprezentabil la atributele de vârfuri, respectiv culoare ale shader-ului şi transmite matricele de transformare în shader-ul de vârfuri pentru a fi aplicate vectorilor de poziţie.



În shader-ul de vârfuri matricele de transformare sunt menţinute într-un *uniform block* şi sunt aplicate vectorului de poziţie în următoarea ordine:

gl_Position = transfBlock.resizeMatrix
**** transfBlock.cameraTranslation***
**** transfBlock.modelTranslation***
**** transfBlock.modelRotation***
**** transfBlock.modelScale * in_Position***

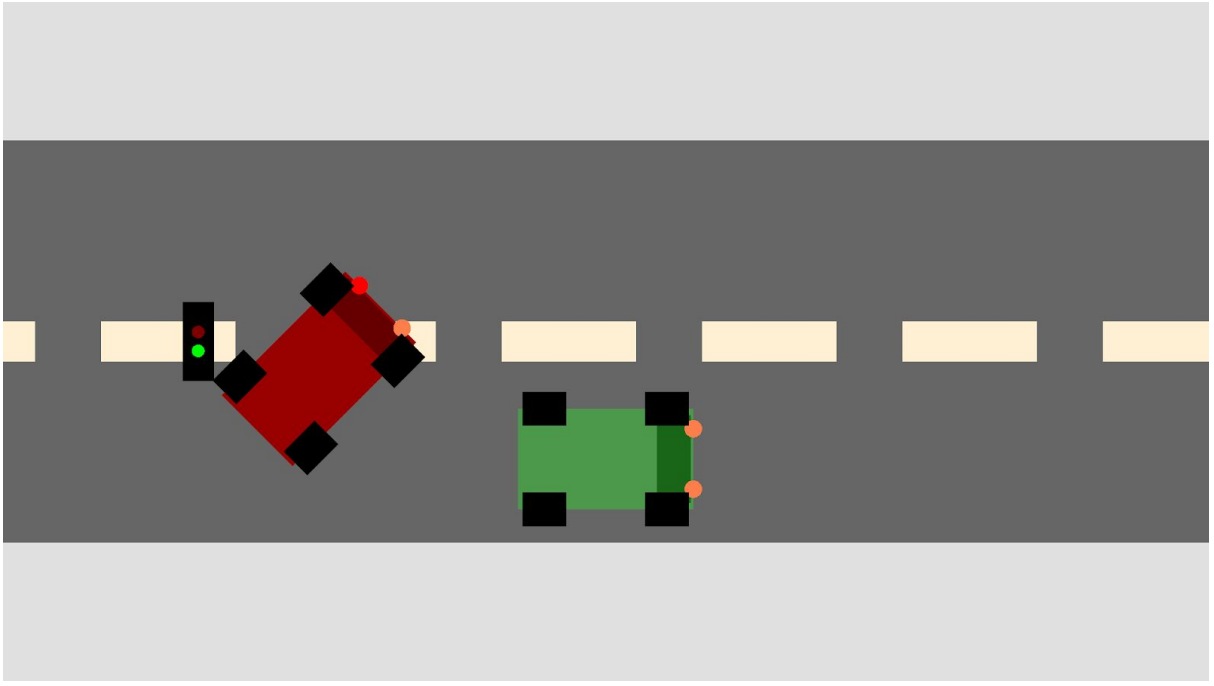
Maşinile sunt alcătuite din mai multe dreptunghiuri (obiecte din clasa ***Rectangle***) pentru reprezentarea caroseriei şi a roţilor şi două discuri (obiecte din clasa **Circle**) folosite ca semnalizatoare, ce se aprind şi se sting intermitent atunci când maşina virează (i.e. atunci când maşina este rotită cu un unghi pozitiv semnalizatorul stâng se aprinde, altfel, cel drept).



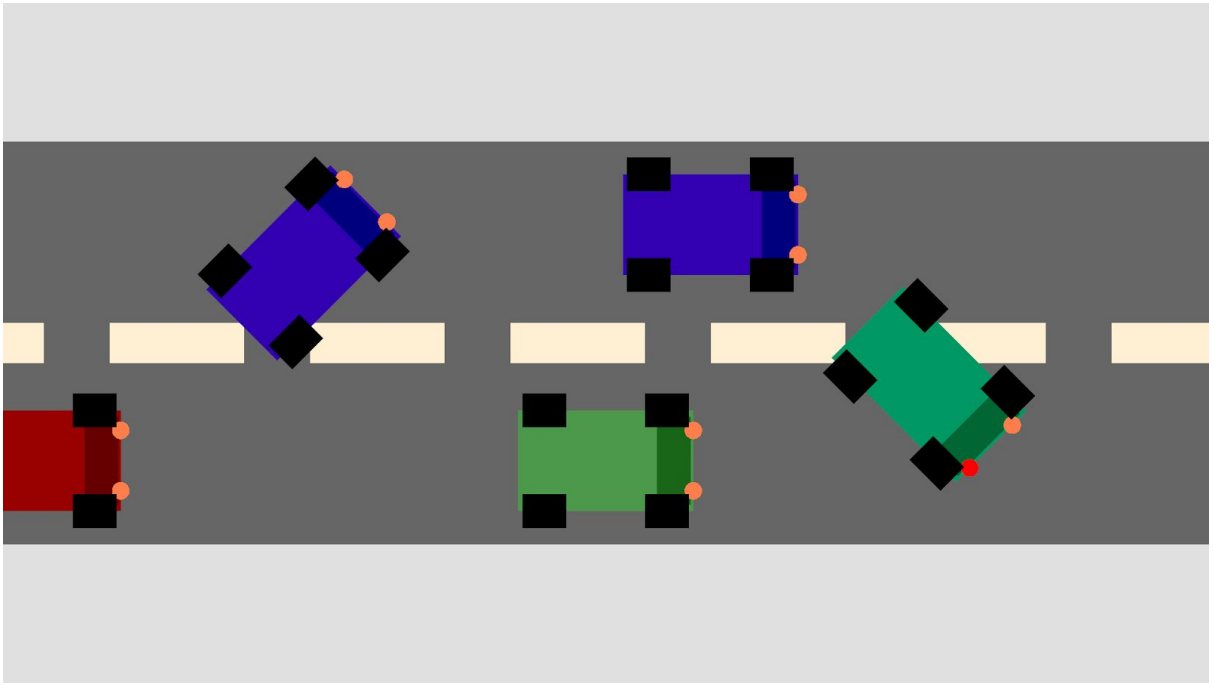
Deplasarea maşinilor începe atunci când culoarea verde a semaforului “se aprinde”. La un interval aleator de timp între 4 şi 10 secunde, o nouă maşină este generată şi adăugată în lista de maşini ce se deplasează independent.
Acestea au o acceleraţie mai mare decât maşina principală (cea a cărei deplasare este urmărită de cameră prin translaţia camerei odată cu maşina). De îndată ce distanţa dintre centrul maşinii urmăritoare şi centrul maşinii urmărite scade sub un anumit prag, maşina urmăritoare începe depăşirea, rotindu-se cu un unghi de Π / 4 şi avansând pe prima bandă până când aceasta găseşte un loc liber pentru a se întoarce pe cea de a doua bandă.

De asemenea, maşinile verifică coliziunea cu celelalte maşini. Dacă maşina principală este “condusă” de către utilizator într-una dintre maşinile independente, simularea se opreşte, maşinile rămânând fixate în poziţia din momentul “accidentului”.
Maşinile ne principale se “împing” între ele în cazul unei coliziuni.

Cadre:



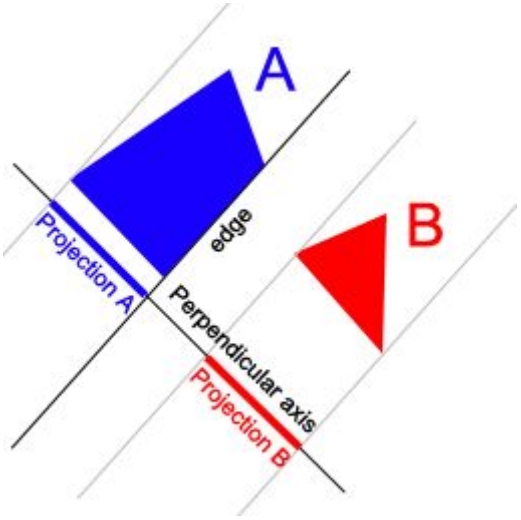
(Începutul primei depăşiri -> lumina verde a semaforului s-a aprins, maşinile au pornit, masina rosie s-a apropiat suficient de masina principala pentru a începe depăşirea)



(Mai multe maşini generate în scenă)

Coliziunea este verificată conform teoremei *“axei separatoare”* (*Separating Axis Theorem*) ce caută o axă perpendiculară pe una dintre muchiile celor două poligoane ale căror coliziune este verificată şi proiectează pentru fiecare poligon vârfurile sale pe axă. Cele două proiecţii ale poligoanelor formează două intervale pe axă. Dacă găsim două intervale de acest tip care nu se suprapun atunci poligoanele nu se intersectează.

Verificarea poate fi simplificată având în vedere ca poligoanele pentru care se testează coliziunea sunt dreptunghiuri deci este suficient să verificăm pentru două muchii din fiecare dreptunghi. De asemenea nu mai este necesar să calculăm perpendiculara pentru o muchie, deoarece cealaltă muchie va fi folosită ca axă perpendiculară pe muchia curentă.



(sursa Code Project: https://www.codeproject.com/Articles/15573/2D-Polygon-Collision-Detection)

Aplicaţia este modularizată, urmărindu-se extensibilitatea şi izolarea funcţionalităţilor distincte în clase separate, identificându-se câteva clase astfel:

- ***ShaderLoader***: clasă ce se ocupă de citirea codului sursă a unui shader, compilare şi eventuala aruncare de excepţii in cazul erorilor în sursele shaderelor
- ***ProgramLoader***: clasă ce leagă mai multe shadere într-un program după ce au fost compilate de către ***ShaderLoader***e
- ***WindowHandler***: abstractizări peste *GLFW* de instanţiere a ferestrelor, adăugare de callback-uri pentru evenimente de tip tastă
- ***Context2D***: desenarea obiectelor ***DrawableObject***, instanţierea şi apelarea metodelor din ***ShaderLoader***, ***ProgramLoader*** şi ***WindowHandler***
- ***DecorationsHandler***: clasă ce se ocupă de desenarea obiectelor din fundal
- ***CarsHandler***: clasă ce se ocupă de desenarea şi poziţionarea maşinilor, verificarea de coliziuni, etc.

Diagrama UML ce descrie arhitectura aplicaţiei: