1. beadandó feladat: egyszerű grafikus felületű alkalmazás

Közös követelmények:

- A megvalósításnak működős programot kell biztosítania. A program elindítását követően egyértelműnek kell lennie, melyik feladatot oldja meg (szöveges információk figyelembe vétele nélkül). A program működése során csak a grafikus képernyőt használhatja (konzolt nem), és billentyűzettel, illetve egérrel lehet vezérelni. A programból bármikor ki lehessen lépni az ESC billentyű segítségével.

- A programkód szerkezetében törekedni kell az egységes programozási stílus követésére. A kódot alprogramok, illetve típusok segítségével részegységekre kell tördelni. *A programkódnak minimum egy saját típust kell használnia*. A típusok megvalósításában törekedni kell az újrafelhasználhatóságra (egységbe zárás, láthatóság kezelése, konstans metódusok, külön fordítási egység). A kódismétlődést kerülni kell.

- A kód legyen megfelelően kommentezett. A főprogram kódfájlja a hallgatói adataival, illetve a program kezelésével kapcsolatos leírással kezdődjön.

- A feltűntetett pontszámok az alapjáték megvalósítására adható pontszámok. További funkciók, kiegészítések, illetve minőségi megoldások növelhetik a maximális pontszámot.

Fogalmak:

- *Képfájl*: egy olyan szövegfájl, ami egy képet tárol úgy, hogy az első két szám, ami a fájlban van az X és az Y mérete a képnek, és utána sorfolytonosan az egyes pixelek szín komponensei vannak R, G, B sorrendben. Egy 100×100-as kép kép fájljában tehát összesen $2 + 100 \times 100 \times 3$ egész szám található.

- *Sprite*: olyan (kisméretű) kép, ami a képernyőn általában változtatja a pozícióját, tipikusan a játékok figurái ilyenek, mint a Mario.

Feladatok:

1. **Békaügető (20 pont)**

   Készítsünk programot egy békaügető verseny szimulációjára.
   A versenyt négy béka játssza, akik a képernyő bal oldalán kezdenek (egymás alatt), és ugrálgatnak jobbra, amíg el nem érik a képernyő jobb szélét. A pálya összesen 20 mezőből áll, amelyen lehetnek gödrök (összesen 5-10 a pályán). Minden béka véletlenszerű időközönként ugrik (1-3 másodperces időintervallumban) 1, vagy 2 mezőt.

Amennyiben gödörbe ugrik, akkor onnan nem tud tovább ugrani. A győztes az a béka, aki legelőször elér a célba. A békákat sprite-okkal jelenítsük meg.

2. Löveg (20 pont)

Készítünk programot, amelyben egy ágyúlöveget irányíthatunk az egér segítségével.
Az ágyú a képernyő bal oldalának alján helyezkedik el, egy célpont pedig a másik oldalon. A játékos feladata a célpont eltalálása. Az ágyú irányát az egér pozíciója szabja meg (az ágyúhoz képest, függetlenül az egér távolságától). Egérkattintásra a lövedék elindul az ágyúból az egér felé, azonban a gravitáció hat rá, így a pályája egy, a földhöz közelítő görbét ír le.
A játékos annyit lőhet, amennyit szeretne, a lövedékek száma végtelen. Ha a lövedék eltalálja a célpontot, automatikusan új játék kezdődik.

3. Vonalrajz (20 pont)

Készítünk programot, amelyben egy csík segítségével tudunk a képernyőre rajzolni.
A csík a képernyő egy adott pontjából indul, folyamatosan halad előre egy rögzített sebességgel, a felhasználó pedig elfordíthatja az irányhoz képest balra, illetve jobbra (90°-kal), továbbá bármikor leállíthatja, és újra elindíthatja a rajzolást a szóköz billentyűre. Amikor a csík áthalad egy olyan pozíción, ahol már volt, akkor az így bezárt területet automatikusan átszínezi véletlenszerű színre. Így a végeredmény egy csíkokkal díszített színes kép lesz, amelyet a felhasználó kimenthet Bitmap (bmp) formátumban.

4. Mozgó mozaik (20 pont)

Készítsünk programot, amellyel egy mozgó mozaikot lehet készíteni.
A képernyő egy adott méretű négyzetrácsot alkot (amelynek méretét a felhasználható adhatja meg két számbillentyű lenyomásával, pl. a 2, 5 lenyomása 25×25 kockát jelent, 0, 9 pedig 9×9-est). Egy négyzetre történő kattintás hatására azok átszíneződnek. A szín a kattintásokkal ciklikusan változnak a fekete, fehér, sárga, piros, zöld és kék színek között. A program így tegye lehetővé különféle mozaik képek megrajzolását. Szóköz lenyomására minden mező legyen újra fekete.
A megrajzolt képet lehessen mozgatni úgy, hogy a képet el lehet tolni mind a négy irányba (fel, le, balra, jobbra). Ilyenkor a kép az adott irányba kezd el mozogni körkörösen (pl. ha balra tolunk, akkor az első oszlop a jobb oldalon jelenik meg) másodpercenként egy sort/oszlopot, ENTER lenyomására pedig megáll.

5. Hisztogram megjelenítés és kiegyenlítés (20 pont)

Készítsünk programot, amely alkalmas képi hisztogram megjelenítésre, és kiegyenlítésre.
A hisztogram olyan, jelen esetben 0-tól 255-ig terjedő skála, amely ábrázolja, hogy adott intenzitásértékből hány pixel található a képen. A program jelenítse meg a hisztogramot grafikusan (piros, zöld, kék) komponensenként oszlopdiagram formájában.
A hisztogram kiegyenlítés tipikusan kontraszttalan képeken végzett optikai javítás, amely annak egyenletességét hivatott növelni. Úgy kell a színeket megváltoztatni, hogy a hisztogram a 0-255 tartományra kiszélesedjen, és minél inkább legyen igaz (kerekítési hibák és egyforma pixelek okozta apró eltérések mellett), hogy az X (ahol X 0-255 közötti érték) világos, és annál sötétebb színekkel rendelkező pixelek száma az összes pixel X/255-öd része, vagyis arányosan oszlik el. Például a kép pixeleinek fele 128-nál sötétebb, negyede 64-nél is sötétebb, stb.

6. Vonal-párbaj (25 pont)

Készítsünk programot, amellyel a következő játékot lehet játszani.
Két játékos játszik egymás ellen. Mindkettő egy-egy vonalat rajzol a képernyőn úgy, hogy a vonal minden másodpercben a legutoljára beállított irányba folytatódik feltéve, hogy a játékos nem változtatja meg azt egy megfelelő billentyű lenyomásával, amely egy új irányt jelöl ki. Az a játékos veszít, aki előbb neki ütközik a másik játékos vonalának vagy a képernyő szélének.
A játék egy 100×100 mezős táblán zajlik, és a vonalak egy mező szélesek, és mindig egy mezőt lépnek előre.

7. Begyűjtés (25 pont)

Készítsünk programot a következő egyszemélyes játékra.
A játékban a felhasználó egy négyzetet irányít, amely mindig az egér aktuális pozíciójában helyezkedik el.
A képernyőre folyamatosan érkeznek a négy irányból további négyzetek (véletlenszerűen), amelyek rögzített (de négyzetenként változó) sebességgel haladnak a képernyő másik végébe (tehát egyenesen végighaladnak a képernyőn). Az érkező négyzetek lehetnek ehetőek, illetve ártalmasak. Ha a játékos négyzete ártalmassal érintkezik, vége a játéknak.
A játékos célja az ehető négyzetek elfogyasztása (érintkezés velük), ekkor azok eltűnnek a képernyőből. Minden fogyasztással a játékos négyzete megnő, így idővel egyre nehezebb elkerülni az ártalmas négyzeteket.
A program folyamatosan jelenítse meg a játékos pontszámát (begyűjtött négyzetek száma).

8. Akadálypálya (25 pont)

Készítsünk programot a következő egyszemélyes játékra.
A játékban a felhasználó egy motort irányít, amely a képernyő alján helyezkedik el, és balra, illetve jobbra lehet dönteni. Ahogy a motor halad a pályán akadályok tűnnek fel, amelyeket ki kell kerülnie, ha egy akadállyal ütközik, vége a játéknak. Az idő előrehaladtával egyre több akadály jelenik meg a pályán, a játékos feladata így minél tovább elkerülni az ütközést. A progam a játék közben folyamatosan jelenítse meg a játékos pontszámát (megtett út).
A motort és az akadályokat jelenítsük meg sprite-ok segítségével.

9. Pingpong játék (25 pont)

Készítünk programot a következő kétszemélyes játékra.
A kép két szélén egy-egy felfelé, illetve lefelé mozgatható csík van. Az egyik csík a kurzorbillentyűkkel, a másik az egérrel vezérelhető. A képen középről véletlen irányba egyenletes sebességgel elindul egy sprite. Ha a falhoz vagy a csíkokhoz ér, akkor visszapattan, ha pedig a csíkok mellett fér el, akkor kiírja középre, hogy a billentyűs vagy az egeres nyerte a meccset, és szóköz leütést vár, majd elölről kezdi.
A program számolja össze a két versenyző által nyert meccseket, és ezt is jelenítse meg az eredménnyel.

10. Szabályozott löveg (25 pont)

Készítünk programot, amelyben egy ágyúlöveget irányíthatunk egér és billentyűzet segítségével.
Az ágyú a képernyő bal oldalának alján helyezkedik el, egy célpont pedig a másik oldalon. A játékos feladata a célpont eltalálása. Az ágyú irányát az egér pozíciója szabja meg (az ágyúhoz képest, függetlenül az egér távolságától). A lövés erősségét billentyűzettel változtathatjuk, és ez változtatja a lövedék haladási sebességét. Egérkattintásra a lövedék elindul az ágyúból az egér felé, azonban a gravitáció hat rá, így a pályája egy, a földhöz közelítő görbét ír le. A gravitáció hatása annál kisebb, minél nagyobb erővel lövünk (de teljesen nem szűntethető meg).
A játékosnak meghatározott számú lövedéke áll rendelkezésre, így ha mind elfogy, és nem találja el a célpontot, vesztett. A program folyamatosan jelenítse meg az ágyú aktuális irányát és erősségét, valamint a megmaradt lövedékek számát.

11. Konvolúciós szűrések (25 pont)

Készítsünk programot, amellyel képi tartalmat szűrhetünk meg konvolúciós alapon, azaz egy páratlan méretű transzformációs mátrix végigcsúsztatásával a felvételen.
A szűrő mátrixa (kernel) adja meg az átalakítás típusát, amely lehet homályosítás (pl. box filter, Gauss filter), élesítés (mean removal, unsharp masking), éldetektálás (pl. Laplace filter, Emboss filter), stb.
Az alkalmazásnak támogatnia kell kép fájlok betöltését, majd rajtuk szűrések végrehajtását. Legalább 5 beépített szűrőt kell tartalmaznia az alkalmazásban, illetve támogatnia kell egyedi szűrők létrehozását (fájlból történő beolvasással, vagy a programban történő megadással). A programnak elég a 3×3-as méretű szűrőket támogatnia.
A kész képet lehessen kimenteni Bitmap (bmp) formátumban.

12. Tetris (30 pont)

Készítsünk programot a közismert Tetris játékra.
Adott egy n×m pontból álló tábla, amely kezdetben üres. A tábla tetejéről egymás után új, 4 kockából álló építőelemek hullanak, amelyek különböző formájúak lehetnek (kocka, egyenes, L alak, tető, rombusz). Az elemek rögzített sebességgel esnek lefelé, és az első, nem telített helyen megállnak. Amennyiben egy sor teljesen megtelik, az eltűnik a játékmezőről, és minden felette lévő kocka eggyel lejjebb esik.
A játékosnak lehetősége van az alakzatokat balra, jobbra mozgatni, valamint forgatni óramutató járásával megegyező irányba, így befolyásolhatja azok mozgását. A játék addig tart, amíg a kockák nem érik el a tábla tetejét.
A program folyamatosan jelenítse meg a játékos pontszámát (eltűntetett sorok száma), és ha vége a játéknak, automatikusan kezdjen újat.

13. Falbontás (30 pont)

Készítsünk programot a következő egyszemélyes játékra.
A játékos egy, a képernyő alján vízszintesen mozgatható ütőt mozgat, amely egy golyó visszapattintására alkalmas, amennyiben eltalálja. A golyó az oldalsó falakról, illetve a tetőről is visszaverődik, de ha eléri a padlót, vége a játéknak. A visszapattanás szöge megegyezik a beérkezés szögével (ellentétes irányba). A pálya felső részén téglák helyezkednek el egy adott formában. Ha a golyó egy téglával ütközik, akkor a tégla eltűnik, maga a golyó pedig visszaütődik.
A játékos célja, hogy lebontsa a téglalapok alkotta halmazt teljes egészében. A téglák alkotta fal formáját beolvashatjuk kész pálya alaprajzokból, vagy generálja véletlenszerűen. A fal lebontása után a program kezdjen automatikusan új játékot.

14. Gyorsulás (30 pont)

Készítsünk programot a következő egyszemélyes játékra.
A játékban a felhasználó egy autót irányít, amely rögzített, de egyre nagyobb sebességgel halad egy kanyargós pályán. A pálya egyes részei egyenesek, de adott időközönként kanyarok is megjelennek, amelyek különböző mértékűek lehetnek (30°, 45°, 90°, 180°). A játékos feladata, hogy az autót balra, illetve jobbra irányítsa, hogy az be tudja venni a kanyarokat, és az úton maradjon. A kanyarodás mértéke attól függ, hogy a játékos mennyi ideig nyomja a billentyűt (az is lehet, hogy túlkanyarodik). Játék közben a program folyamatosan jelenítse meg a játékos pontszámát (megtett út). Ha az autó letért az útról, a játéknak vége.
A levezetendő utat betölthetjük fájlból, vagy generálhatjuk véletlenszerűen.
Az autót jelenítsük meg sprite segítségével.

15. Aknamező (30 pont)

Készítsünk programot a következő egyszemélyes játékra.
A játékban egy tengeralattjárót kell irányítanunk a képernyőn (balra, jobbra, fel, illetve le), amely felett ellenséges hajók köröznek, és folyamatosan aknákat dobnak a tengerbe. Az aknáknak három típusa van (könnyű, közepes, nehéz), amely meghatározza, hogy milyen gyorsan süllyednek a vízben (minél nehezebb, annál gyorsabban).
Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna. Az egyes játékbeli elemeket jelenítsük meg sprite-ok segítségével.
A program ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő.

16. Tankcsata (30 pont)

Készítsünk programot a következő kétszemélyes játékra.
A játékban egy-egy tank szerepel, amelyek a képernyő két oldalán helyezkednek el (oldalról nézve), köztük pedig hegyek találhatóak (amelyeket nem lehet átlőni). A játékosok feladata, hogy a másik tankot kilőjék úgy, hogy felváltva lőhetnek. A lövésnél szabályozható az irány, illetve az erősség. Az erősség befolyásolja a lövedék haladási sebességét. A lövedékre hat a gravitáció, így a pályája egy, a földhöz közelítő görbét ír le. A gravitáció hatása annál kisebb, minél nagyobb erővel lőjük ki a lövedéket (de teljesen nem szűntethető meg). A játéknak akkor van vége, ha egy löveg valamelyik tankhoz igen közel csapódik be (egy függőleges lövéssel a játékos saját magát lövi le).

17. Geometriai transzformációk (30 pont)

Készítsünk programot, amely alkalmas képfájlok betöltésére, és rajta az alábbi képszerkesztési eljárások elvégzésére:

- elforgatás szögenként, egér húzásával, vagy iránygombokkal vezérelve,
- tükrözés (pontosabban megfordítás) vízszintesen, illetve függőlegesen (a középvonalra),
- nagyítás, illetve kicsinyítés egérgörgővel vezérelve (százalékonként),

Az egyes transzformációkhoz a képet újra kell méretezni, és az eredi képről kettős lineáris (bilinear) mintavételezés segítségével leképezni a képpontokat.
Legyen lehetőség a keletkezett kép elmentésére Bitmap formátumban.