DUM č. 12 v sadě

35. Inf-11 Objektové programování v Greenfoot

Autor: Lukáš Rýdlo

Datum: 30.06.2014

Ročník: studenti semináře

Anotace DUMu: Podpora myši ve hře. Vykreslení lišty ikon a reakce na kliknutí. Vkládání hráčů myší.

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Projekt: Simulace života zajíce v C/Allegro
Ovládání myší

Úvod

Náš projekt jsme zatím ovládali pouze pomocí klávesnice. Bývá zvykem, že grafické aplikace se ovládají spíše myší, takže by bylo vhodné tuto funkcionalitu implementovat. Umožní nám to například vkládat nové hráče na konkrétní místo a nejen do středu plánu.

Pro rozumné ovládání zavedeme lištu s ikonami, které budou reagovat na kliknutí. Také zavedeme možnost vypnout aplikaci křížkem okna a nastavení titulku okna.

Úkoly s řešeními

1. Vytvořte funkci drawIcons, která bude přebírat adresu na buffer displeje a pole obrázků hráčů a vykreslí k pravému okraji obrazovky svisle pod sebe čtvercová tlačítka s následujícími obrázky: první obsahuje zelený trojúhelníček symbolizující spuštění simulace nebo dva svislé obdélníčky jako pauzu podle stavu globální proměnné simulationPause, druhé obsahuje červený křížek (tato dvě tlačítka kreslete funkcemi kreslení grafických primitiv), další obrázek zajíce, zaječky, lišky a mrkve (použijte načtené obrázky z pole). Všechna tlačítka ať jsou rozumně a stejně velká, jejich velikost je daná direktivou ICON_SIZE, obrázky budete muset asi zmenšovat vhodnou funkcí.

   Úkol je principiálně vzato jen „řemeslné kreslení“, pouze je nutné najít si správné kreslící funkce mezi funkcemi pro kreslení primitiv a také pro vykreslení obrázku bude nutné najít si funkci, která umí „sprite“ vykreslit zmenšený. Zdůrazňuji nutnost počítat správně souřadnice tlačítek a rozměry kreslených útvarů tak, aby při změně konstanty ICON_SIZE na libovolnou (kladnou) hodnotu byl obrázek korektně vykreslen, včetně mezer mezi tlačítky. Dávejte proto pozor na dělení celými čísly. Kvůli zaokrouhlování a celočíselnému dělení bývá obvykle výhodnější dělit desetinnou konstantou, aby se výpočet provedl s přesností racionálních čísel a pak teprve zkonvertoval. Nezapomeňte také při zmenšování obrázků spočítat poměr stran a zmenšovat obrázek se zachováním poměru stran. Tím se sice výpočet mírně komplikuje, ale není to tak náročné a výsledek vypadá o poznání lépe, než pokud se obrázek deformuje.

```
void drawIcons(□ buffer, □ images) {
   int x = SCREEN_□-ICON_SIZE*□/□;
   int y = ICON_SIZE/□;
   int i = 0;
   □ scale;

   for (i = 0; i<6; □, y□3/2.0*ICON_SIZE) {
      □(buffer,x,y,x+ICON_SIZE,y+ICON_SIZE,make□(255,150,0));
      □(buffer,x+2,y+2,x+□-2,y+□-2,make□(255,255,150));

      switch(i) {
         case 0:
            if(simulationPause) {
               □(buffer,x+6,y+6,x+6,y+□-6,x+□-6,y+□/2,make□(0,150,0));
            } else {
               □(buffer, x+□/□.0,y+□,x+□*□/□.0,y+□-□,make□(0,150,0));
               □(buffer, x+□*□/□.0,y+□,x+□*□/□.0,y+□-make□(0,150,0));
            }
            break;
         case 1:
            □(buffer,x+□,y+□,x+□-□,y+□-□,make□(150,0,0));
            □(buffer,x+□,y+□,x+□-□,y+□-□,make□(150,0,0));
            □(buffer,x+□,y+□,x+□-□,y+□-□,make□(150,0,0));
            □(buffer,x+□,y+□-□,x+□-□,y+□,make□(150,0,0));
            □(buffer,x+□,y+□-□,x+□-□,y+□,make□(150,0,0));
            break;
         case 2:
```

```
                scale = images[AT_RABBIT]->□/images[AT_RABBIT]->□;
                □_sprite(buffer, □, x+□, y+(□+□-((□-□)*□))/□.0,
                         □-□, (□-□)*□);
                break;
            case 3:
                scale = images[AT_FRABBIT]->□/images[AT_FRABBIT]->□;
                □_sprite(buffer, □, x+□, y+(□+□-((□-□)*□))/□.0,
                         □-□, (□-□)*□);
               break;
            case 4:
                scale = images[AT_FOX]->□/images[AT_FOX]->□;
                □_sprite(buffer, □, x+□, y+(□+□-((□-□)*□))/□.0,
                         □-□, (□-□)*□);
               break;
            case 5:
                scale = images[AT_CARROT]->□/images[AT_CARROT]->□;
                □_sprite(buffer, □, x+□, y+(□+□-((□-□)*□))/□.0,
                         □-□, (□-□)*□);
               break;
        }
    }

}
```

2. Zapněte zobrazování kurzoru, použijte rozhraní Allegra a jeho vlastní kurzory. Výchozí kurzor je šipka.

   Dohledejte v dokumentaci obě funkce, které musíte zavolat (nastavení kurzoru a vykreslení). Inicializace je již zapnutá z demo kódu ve funkci init.

3. Implementujte funkcionalitu tlačítek. Napište funkci checkMouse, která bude přebírat obrázky hráčů. Detekuje stisk levého tlačítka myši nad některým tlačítkem z lišty naprogramované v minulém úkolu. Pozastaví nebo rozběhne simulaci, u křížku dealokuje všechny hráče (opatrně s přerušením, během dealokace je rozumné časovač zrušit a teprve po dealokaci jej obnovit). V případě ikon s hráči nastaví obrázek kurzoru na obrázek hráče a nastaví globální proměnnou addActor tak, aby při příštím kliknutí kamkoliv v herním plánu na toto místo vložila zvoleného hráče. Při kliku pro vložení hráče nezapomeňte opět obnovit původní obrázek kurzoru – šipku. Uvědomte si, jak Allegro pracuje s detekcí stisku myši – je to podobné jako u klávesnice, takže nejspíš bude zapotřebí zavést mouseDelay a MOUSE_DELAY tak, jako keyDelay a KEY_DELAY…

   K implementaci kliknutí na tlačítka využijte stejný cyklus jako u vykreslování tlačítek, jen místo kreslení detekujte, zda je kurzor v mezích hranic x a y. Při vkládání nového hráče testujte, zda globální proměnná addActor obsahuje hodnotu, která patří hráči. Máme k tomu hotovou direktivu… Při spuštění programu je potřeba, aby hodnota addActor byla nastavena tak, že se nic nevkládá.

```
void checkMouse(□ image) {
   int x = SCREEN_□-ICON_SIZE*□/□;
   int y = ICON_SIZE/□;
   int i = 0;
```

```
    if(□>0) return;

    if (mouse_□ □ □) {

        if(□(addActor)) {
            createActor(□,mouse_□+□->□/□,mouse_□+□->□/□,(□!=□?2:0),0);
            addActor=□;
            □(NULL);
        }

        for (i = 0; i<6; □, y□3/2.0*ICON_SIZE) {
            if (mouse_□ □ □ □ mouse_□ □ □+□ &&
                mouse_□ □ □ □ mouse_□ □ □+□) {

                mouseDelay=□;

                switch(i) {
                    case 0:
                        simulationPause□simulationPause;
                        break;
                    case 1:
                        simulationPause=□;
                        □(timeInterrupt);
                        deallocActors();
                        □(timeInterrupt,10);
                        break;
                    case 2:
                        □(image[AT_RABBIT]);
                        addActor = □;
                        break;
                    case 3:
                        □(image[AT_FRABBIT]);
                        addActor = □;
                        break;
                    case 4:
                        □(image[AT_FOX]);
                        addActor = □;
                        break;
                    case 5:
                        □(image[AT_CARROT]);
                        addActor = □;
                        break;
                }
            }
        }
    }
}
```

4. Nastavte titulek okna na „Rabbit simulation“. Zajistěte možnost vypnout aplikaci kliknutím na křížek okna (zatím nefunguje). Hledejte v dokumentaci pod kapitolou Using Allegro.

   K řešení stačí použít přímo ukázkový kód z dokumentace Allegro. Nastavení titulku i callbacku pro vypnutí se provede ve funkci init. Všimněte si, co je to callback – je to funkce předaná jiné funkci jako parametr. Tato funkce je pak „zavolaná zpět“ („called back“) ve chvíli vykonání nějaké akce. Toto je typické chování pro grafické aplikace a přístup, kterému se říká událostmi řízené programování. V nekonečné smyčce programu se hlídá vznik události (kliknutí na křížek) aby se vykonala nějaká akce (zavolal callback).

Úkoly

1. Vytvořte funkci drawIcons, která bude přebírat adresu na buffer displeje a pole obrázků hráčů a vykreslí k pravému okraji obrazovky svisle pod sebe čtvercová tlačítka s následujícími obrázky: první obsahuje zelený trojúhelníček symbolizující spuštění simulace nebo dva svislé obdélníčky jako pauzu podle stavu globální proměnné simulationPause, druhé obsahuje červený křížek (tato dvě tlačítka kreslete funkcemi kreslení grafických primitiv), další obrázek zajíce, zaječky, lišky a mrkve (použijte načtené obrázky z pole). Všechna tlačítka ať jsou rozumně a stejně velká, jejich velikost je daná direktivou ICON_SIZE, obrázky budete muset asi zmenšovat vhodnou funkcí.

2. Zapněte zobrazování kurzoru, použijte rozhraní Allegra a jeho vlastní kurzory. Výchozí kurzor je šipka.

3. Implementujte funkcionalitu tlačítek. Napište funkci checkMouse, která bude přebírat obrázky hráčů. Detekuje stisk levého tlačítka myši nad některým tlačítkem z lišty naprogramované v minulém úkolu. Pozastaví nebo rozběhne simulaci, u křížku dealokuje všechny hráče (opatrně s přerušením, během dealokace je rozumné časovač zrušit a teprve po dealokaci jej obnovit). V případě ikon s hráči nastaví obrázek kurzoru na obrázek hráče a nastaví globální proměnnou addActor tak, aby při příštím kliknutí kamkoliv v herním plánu na toto místo vložila zvoleného hráče. Při kliku pro vložení hráče nezapomeňte opět obnovit původní obrázek kurzoru – šipku. Uvědomte si, jak Allegro pracuje s detekcí stisku myši – je to podobné jako u klávesnice, takže nejspíš bude zapotřebí zavést mouseDelay a MOUSE_DELAY tak, jako keyDelay a KEY_DELAY…

4. Nastavte titulek okna na „Rabbit simulation“. Zajistěte možnost vypnout aplikaci kliknutím na křížek okna (zatím nefunguje). Hledejte v dokumentaci pod kapitolou Using Allegro.