Contents

Ohranjevalniki zaslona

Miha Maksimiljan Bertoncelj

January 6, 2025

1 Uvod

Ohranjevalniki zaslona so bili prvotno zasnovani za odpravo težave trajnih zapečenih slik (angl. phosphor burn-in) na zgodnjih CRT in plazma zaslonih. Čeprav sodobni zasloni to težavo ne poznajo, ohranjevalniki zaslona še naprej izpolnjujejo estetske, varnostne in uporabniške funkcije.

Ta seminar preučuje ohranjevalnike zaslona s perspektive sistemsko neodvisnih principov, specifičnosti implementacije za različne operacijske sisteme ter tehničnih pristopov k razvoju. Na koncu je predstavljen primer preprostega ohranjevalnika zaslona, zasnovanega za macOS.

2 Opis problema

Pred pojavom LCD zaslonov je večina računalniških zaslonov temeljila na katodnih cevkah (angl. cathode-ray tubes, CRT). Pri dolgotrajnem prikazu iste slike na CRT zaslonu se lastnosti fosforne prevleke na notranji strani zaslona postopoma in trajno spremenijo, kar vodi do potemnelih senc (angl. ghost image) na zaslonu, poimenovanih kot zapečena slika (angl. screen burn-in).

Programi ohranjevalnikov zaslona so bili zasnovani z namenom preprečevanja teh učinkov, saj samodejno spreminjajo slike na zaslonu med obdobji neaktivnosti uporabnika.

Pri CRT napravah, ki so v javni uporabi, kot so bankomati in drugi avtomati, je tveganje za zapečeno sliko še posebej visoko, ker se v stanju pripravljenosti na zaslonu pogosto prikazuje statična slika. V teh primerih popolno zatemnitev zaslona ni izvedljiva, saj bi uporabniki napravo lahko zaznali kot nedelujočo. Zapečenje se lahko prepreči z manjšimi premiki vsebine zaslona vsakih nekaj sekund ali z uporabo več različnih slik, ki se redno spreminjajo.

Za preprečevanje zapečenja slik se uporablja tudi tehnika premikanja pikslov (angl. pixel shifting), ki občasno premakne prikazano vsebino za nekaj pikslov, s čimer zmanjša tveganje za trajne poškodbe na zaslonu.

3 Sistemska logika

Osnovni koncept ohranjevalnikov zaslona zajema naslednje ključne funkcionalnosti:

- Aktivacija po določenem obdobju uporabnikove neaktivnosti.
- Zaznavanje vnosa s tipkovnice ali miške za deaktivacijo.
- Dinamična ali statična vizualizacija grafičnih elementov.

Učinkovitost ohranjevalnika temelji na optimizirani rabi strojnih in programskih virov, kar omogoča nemoteno delovanje brez motenja drugih procesov.

Za aktivacijo in prekinitev ohranjevalnika in zanavanje vnosa običajno skrbi screen saver engine ali applikacija, ki je ze namescena kot del operacijskega sistema. Ta nato zažene dejanski ohranjevalnik zaslona. ***(povemo v kako se prevede apa neki)*** Nekaj ohranjevlanikov zaslona je po navadi že del operacojskega sistema. Preko grafičnega umesnika jih lahko izbiramo in kofiguriramo.

4 Specifikacije operacijskih sistemov

4.1 Windows

Microsoft Win32 API podpira posebne aplikacije, imenovane ohranjevalniki zaslona. Ko je ohranjevalnik zaslona izbran, Windows spremlja pritiske tipk in premike miške ter po obdobju neaktivnosti zažene ohranjevalnik zaslona. Windows ne zažene ohranjevalnika zaslona, če obstajajo naslednji pogoji:

- Aktivna aplikacija ni aplikacija, zasnovana za Windows.
- Prisotno je okno za računalniško usposabljanje (CBT).
- Aktivna aplikacija prejme sporočilo `WM_SYSCOMMAND` z nastavljeno vrednostjo parametra wParam na `SC_SCREENSAVE`, vendar sporočila ne posreduje funkciji DefWindowProc.

Da bi ugotovil, ali lahko zažene zaščito zaslona ali ne, Windows pošlje sporočilo aplikaciji, ki je v ospredju. Ta ukaz aplikacijo vpraša: "Ali lahko zažgem zaščito zaslona?" Programi, ki niso za Windows, tega ukaza ne bodo razumeli, zato nanj ne bodo odgovorili. CBT aplikacija ga bo razumela, vendar bo odgovorila z ukazom, ki pomeni: "Ne, trenutno nudim usposabljanje." Vse druge aplikacije bi morale odgovoriti pozitivno na ukaz.

Windows nato pogleda vrstico `SCRNSAVE.EXE=____` v datoteki system.ini, da preveri, ali je zaščita zaslona določena. Če je vnos prazen, ukaz za zagon zaščite zaslona ignorira. Če pa je naveden ime datoteke, poskusi naložiti to datoteko. Dokler je datoteka, ki je

navedena, dejanska zaščita zaslona, program izvede in ustvari slike zaščite zaslona na vrhu trenutnega namizja.

Knjižnica ohranjevalnikov zaslona vsebuje glavno funkcijo in drugo zagonsko kodo, potrebno za ohranjevalnik zaslona. Ko se ohranjevalnik zaslona zažene, zagonska koda v knjižnici ohranjevalnika ustvari okno čez celoten zaslon. Razred okna za to okno je deklariran na naslednji način:

```
WNDCLASS cls;
cls.hCursor        = NULL;
cls.hIcon          = LoadIcon(hInst, MAKEINTATOM(ID_APP));
cls.lpszMenuName   = NULL;
cls.lpszClassName  = "WindowsScreenSaverClass";
cls.hbrBackground  = GetStockObject(BLACK_BRUSH);
cls.hInstance      = hInst;
cls.style          = CS_VREDRAW  | CS_HREDRAW | CS_SAVEBITS | CS_DBLCLKS;
cls.lpfnWndProc    = (WNDPROC) ScreenSaverProc;
cls.cbWndExtra     = 0;
cls.cbClsExtra     = 0;
```

4.1.1 Tri zahtevane funkcije

Vsak ohranjevalnik zaslona mora implementirati naslednje tri funkcije, ki so povezane z `screen saver library`
Funkcije so: `ScreenSaverProc ScreenSaverConfigureDialog  RegisterDialogClasses`

ScreenSaverProc Ta funkcija obdeluje specifična sporočila in vsa neobdelana sporočila posreduje knjižnici ohranjevalnika zaslona. Nekatera običajna sporočila, ki jih ScreenSaverProc obdeluje, so:

- WM_CREATE: Pridobi inicializacijske podatke iz datoteke Regedit.ini, nastavi časovnik za okno ohranjevalnika zaslona in izvede druge inicializacije.
- WM_ERASEBKGND: Počisti ozadje okna ohranjevalnika zaslona in pripravi na nadaljnje risanje.
- WM_TIMER: Izvede risalne operacije.
- WM_DESTROY: Uniči časovnike, ustvarjene med obdelavo sporočila `WM_CREATE`, in izvede dodatno čiščenje.

Tako kot večina postopkov za obdelavo oken tudi ScreenSaverProc obdeluje niz specifičnih sporočil in neobdelana sporočila posreduje privzetemu postopku. Vendar pa namesto da bi jih posredoval funkciji DefWindowProc, jih ScreenSaverProc posreduje funkciji DefScreenSaverProc.

Druga razlika med ScreenSaverProc in običajnim postopkom okna je v tem, da ročaj (handle), ki ga posreduje `ScreenSaverProc`, identificira celotno namizje, namesto odjemalskega okna. Naslednji primer prikazuje postopek okna `ScreenSaverProc` za vzorčno zaščito zaslona.

Neobdelana sporočila posreduje knjižnici ohranjevalnikov z uporabo funkcije `DefScreenSaverProc`. Naslednja tabela opisuje, kako ta funkcija obdeluje različna sporočila:

- WM_SETCURSOR: Nastavi kazalec na prazen kazalec in ga odstrani z zaslona.
- WM_PAINT: Nariše ozadje zaslona.
- WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN, WM_KEYDOWN
  Konča ohranjevalnik zaslona.
- WM_ACTIVATE: Konča ohranjevalnik zaslona, če je parameter `wParam` nastavljen na `FALSE`.

ScreenSaverConfigureDialog Ta funkcija prikaže pogovorno okno, ki omogoča uporabniku konfiguracijo ohranjevalnika zaslona (aplikacija mora zagotoviti ustrezno predlogo za pogovorno okno). Windows prikaže konfiguracijsko pogovorno okno, ko uporabnik v nadzorni plošči izbere gumb »Setup« v razdelku za ohranjevalnik zaslona.

RegisterDialogClasses To funkcijo morajo klicati vse aplikacije za ohranjevalnike zaslona. Aplikacije, ki ne potrebujejo posebnih oken ali prilagojenih kontrol v konfiguracijskem pogovornem oknu, lahko preprosto vrnejo TRUE. Aplikacije, ki potrebujejo posebna okna ali prilagojene kontrole, morajo s to funkcijo registrirajo ustrezne razrede oken.
Poleg ustvarjanja modula, ki podpira tri opisane funkcije, mora ohranjevalnik zaslona vključevati tudi ikono. Ta ikona je vidna le, ko se ohranjevalnik zažene kot samostojna aplikacija. (Da se ohranjevalnik zažene prek nadzorne plošče, mora imeti pripono datoteke .scr; za zagon kot samostojna aplikacija mora imeti pripono .exe.) Ikona mora biti identificirana v viru ohranjevalnika zaslona z uporabo konstante `ID_APP`, ki je definirana v glavi `Scrnsave.h`.
Zadnja zahteva je opisna nizovna vrednost ohranjevalnika zaslona. Datoteka virov za ohranjevalnik mora vsebovati niz, ki ga nadzorna plošča prikaže kot ime ohranjevalnika. Opisna vrednost mora biti prvi niz v tabeli nizov datoteke virov (identificirana z ordinalno vrednostjo 1). Če ima ohranjevalnik dolgo ime datoteke, nadzorna plošča opisne vrednosti ignorira in namesto nje uporabi ime datoteke.
Statične funkcije zaščite zaslona so vsebovane v knjižnici zaščite zaslona. Na voljo sta dve različici knjižnice, Scrnsave.lib in Scrnsavw.lib. Vaš projekt morate povezati z eno izmed teh. Scrnsave.lib se uporablja za zaščite zaslona, ki uporabljajo ANSI znake, medtem ko se Scrnsavw.lib uporablja za zaščite zaslona, ki uporabljajo Unicode znake. Zaščita zaslona, povezana s Scrnsavw.lib, bo delovala samo na Windows platformah, ki podpirajo

Unicode, medtem ko bo zaščita zaslona, povezana s Scrnsave.lib, delovala na vseh Windows platformah.

4.2 Varnostni kontekst

Varnostni kontekst ohranjevalnika zaslona je odvisen od tega, ali je uporabnik interaktivno prijavljen. Če je uporabnik interaktivno prijavljen ob zagonu ohranjevalnika zaslona, se ta izvaja v varnostnem kontekstu interaktivnega uporabnika. Če uporabnik ni prijavljen, je varnostni kontekst odvisen od različice operacijskega sistema Windows.

Ko Windows prejme ukaz, da naj konča zaščito zaslona, preveri, ali je zaščita z geslom omogočena. Če je, se prikaže okno, v katerem morate vnesti uporabniško ime in geslo. V nasprotnem primeru se zaščita zaslona preprosto zaključi. Ko je zaščita z geslom aktivirana, neuspešno vnos pravilnega imena in gesla povzroči, da Windows nadaljuje z izvajanjem programa zaščite zaslona. Čeprav to zagotavlja določeno varnost, je pomembno opozoriti, da zaščite zaslona v sistemih Windows 95/98 ustvarijo lastna okna za geslo in zahtevajo geslo ter uporabniške informacije iz sistema. Hekerji lahko in dejansko ustvarijo zaščite zaslona, ki izkoriščajo to šibko točko v varnosti sistema za zajemanje gesel. To ni težava pri sistemih, ki uporabljajo Windows NT, saj ti omogočajo, da zaščite zaslona le pokličejo sistemsko okno za geslo – ne morejo ustvariti svojih lastnih.

5 Linux

XScreenSaver se pogosto uporablja v prostih in odprtokodnih operacijskih sistemih, ki uporabljajo X Window System, kot sta Linux in FreeBSD. Na teh sistemih je na voljo več paketov: eden za okvir za ohranjevanje in zaklepanje zaslona ter dva ali več za prikazne načine.
Na sistemih Macintosh XScreenSaver deluje z vgrajenim ohranjevalnikom zaslona v macOS. Na napravah z iOS je XScreenSaver samostojna aplikacija, ki lahko polnozaslonsko izvaja katerikoli ohranjevalnik. Na napravah z Androidom pa prikazni načini XScreenSaver delujejo kot običajni ohranjevalniki zaslona (ki jih Android včasih imenuje "Daydreams") ali kot žive ozadja.
Uradne različice za Microsoft Windows ni, avtor pa odvrača od poskusov prenosa na to platformo zaradi osebnih zamer do Microsofta in njihovih poslovnih praks.

5.1 Programska arhitektura

Demon XScreenSaver je odgovoren za zaznavanje neaktivnosti, zatemnitev in zaklepanje zaslona ter zagon načinov prikaza. Načini prikaza (imenovani "hacks" zaradi zgodovinske uporabe izraza "display hack") so vsak samostojni programi.
To je pomembna varnostna lastnost, saj so načini prikaza zaprti v ločenem procesu od okvira za zaklepanje zaslona. To pomeni, da napaka v enem od grafičnih načinov prikaza

ne more ogroziti samega zaklepa zaslona (npr. sesutje v načinu prikaza ne bo odklenilo zaslona).
Prav tako to pomeni, da je mogoče napisati ohranjevalnik zaslona tretje osebe v kateremkoli jeziku ali z uporabo katerekoli grafične knjižnice, če je le sposoben risati na zunanji okno, ki je na voljo.
Iz zgodovinskih in prenosljivostnih razlogov so vključeni "hacks" vsi napisani v ANSI C. Približno polovica jih uporablja X11 API, približno polovica pa uporablja OpenGL 1.3 API.
Možnosti za xscreensaver so shranjene na enem od dveh mest: v datoteki .xscreensaver v uporabnikovi domači mapi ali v X bazi virov. Če datoteka .xscreensaver obstaja, prepiše vse nastavitve v bazi virov.
Za nastavitev privzetih vrednosti za celoten sistem je potrebno spremeniti datoteko xscreensaver app-defaults, ki je nameščena ob namestitvi xscreensaver. Ta datoteka je običajno shranjena v /usr/lib/X11/app-defaults/XScreenSaver, vendar je lahko na različnih sistemih shranjena tudi na drugih mestih, na primer /usr/openwin/lib/app-defaults/XScreenSaver na sistemu Solaris.
Ko se nastavitve spremenijo v pogovornem oknu Preferences (nastavitve), se trenutne nastavitve shranijo v datoteko .xscreensaver. Datoteki .Xdefaults in app-defaults pa xscreensaver nikoli ne spremeni ali zapiše.

5.2 Kako deluje?

Ko je čas za aktivacijo ohranjevalnika zaslona, se na vsakem zaslonu prikaže črno okno, ki pokrije celoten zaslon. Vsako okno je ustvarjeno tako, da bo za kasneje ustvarjene programe videti kot "virtualni korenski" (root) zaslon. Zaradi tega lahko vsak program, ki riše na korenskem oknu (in razume virtualne korenske zaslone), deluje kot ohranjevalnik zaslona. Različni grafični prikazi so v resnici samo samostojni programi, ki vedo, kako risati na dodeljenem oknu.
Ko uporabnik ponovno postane aktiven, so okna ohranjevalnika zaslona odstranena, procesi, ki so v teku, pa so prekinjeni z uporabo signala SIGTERM. Tako se procesi prekinejo tudi, ko ohranjevalnik zaslona odloči, da je čas za zagon drugega prikaza: stari proces je končan, novi pa zagnan.

5.3 Varnost

demon XScreenSaver načine prikaza zapre v ločen proces in se poveže s čim manj knjižnicami. Zlasti se ne povezuje z grafičnimi okviri, kot sta GTK ali KDE, ampak uporablja zgolj surovi Xlib za risanje pogovornega okna za odklepanje zaslona.
V zadnjih letih so nekatere distribucije Linuxa začele privzeto uporabljati ogrodja za zatemnitev zaslona, kot sta gnome-screensaver ali kscreensaver, namesto ogrodja, ki je vključeno v XScreenSaver. Leta 2011 je bil gnome-screensaver razdeljen na mate-screensaver in

cinnamon-screensaver. Prejšnje različice teh ogrodij so še vedno uporabljale zbirko ohranjevalnikov zaslona XScreenSaver, kar je več kot 90 % paketa. Vendar pa je v letu 2011 različica gnome-screensaver 3 popolnoma opustila podporo za ohranjevalnike zaslona in podpirala zgolj preprosto zatemnitev zaslona, in od leta 2018 različica cinnamon-screensaver 4.0.8 v Linux Mintu ne podpira več XScreenSaver hacks.
Tiste distribucije Linuxa, ki so nadomestile XScreenSaver z drugimi ogrodji za zaklepanje zaslona, so imele opazne varnostne težave. Ta druga ogrodja imajo zgodovino varnostnih napak, ki omogočajo odklepanje zaslona brez gesla, na primer, preprosto držanje tipke, dokler se zaklepanje ne sesuje.
Leta 2004 je Zawinski pisal o arhitekturnih odločitvah, sprejetih v XScreenSaver, s ciljem izogibanja tovrstnim napakam, zaradi česar je leta 2015 pripomnil: "Če na Linuxu ne uporabljate XScreenSaver, potem lahko z gotovostjo trdite, da vaš zaslon ni zaklenjen."

6 macOs

Ohranjevalnik zaslona na macOS-u deluje kot vtičnik za pogon ohranjevlanika zaslona, ki se lahko integrira v sistem s pomočjo paketa z pripono .saver. Ta paket se namesti v enega od imenikov `Library/Screen Savers` v sistemu. V okviru paketa je vključena izvršna datoteka, ki vsebuje podrazred `ScreenSaverView`. Ta podrazred omogoča definiranje vmesnika, ki je uporabljen za ustvarjanje vsebine ohranjevalnika zaslona. Če ohranjevalnik zaslona shranjuje nastavitve, se namesto standardnega razreda `UserDefaults` uporablja razred `ScreenSaverDefaults`.

Ker so ohranjevalniki zaslona vtičniki za pogon ohrajevalnikov zaslona, mora biti binarna datoteka zaščite zaslona združljiva z arhitekturo strojne opreme, ki jo uporablja trenutni pogon sistema. Pogon zaščite zaslona uporablja naravno arhitekturo gostiteljskega računalnika, kar pomeni, da mora zaščita zaslona podpirati tako arhitekture x86_64 kot arm64 za popolno združljivost.

Ko macOS zažene zaščito zaslona, se izvede naslednji postopek:

- Zaslon počasi zatemni.
- Sistem ustvari primer podrazreda `ScreenSaverView` in pokliče njegovo metodo `init(frame:isPreview:)`, da inicializira zaslon zaščite zaslona.
- Ustvari se okno, v katerem je nameščen podrazred `ScreenSaverView`.
- Okno se aktivira in njegov red se nastavi.
- Pokliče se metoda `draw(_:)` za risanje začetnega stanja ohranjevalnika zaslona.
- Zaslon se počasi osvetli, da razkrije okno ohranjevlanika zaslona na sprednji strani.

- Sistem pokliče metodo `startAnimation()`, ki omogoči nastavitev informacij o stanju animacije.
- Metoda `animateOneFrame()` se večkrat pokliče, da se izriše slika animacije v ohranjevlniku zaslona.
- Ko uporabnik izvede neko dejanje, sistem pokliče metodo `stopAnimation()`, ki ustavi zaščito zaslona. V tej metodi se običajno počistijo vse informacije o stanju, ki so bile nastavljene v metodi `startAnimation()`.
- Opomba: Metodi `stopAnimation()` in `startAnimation()` ne zaženeta ali ustavita animacij takoj. Sistem lahko še vedno pokliče vašo metodo `animateOneFrame()` po tem, ko pokliče `stopAnimation()`.

6.1 koda mojega ohranjevalnika zaslona

```
#import "myscreensaverView.h"

@implementation myscreensaverView {
    NSPoint ballPosition;  // Current position of the ball
    NSSize ballVelocity;   // Velocity of the ball
    CGFloat ballRadius;    // Radius of the ball
}

- (instancetype)initWithFrame:(NSRect)frame isPreview:(BOOL)isPreview
{
    self = [super initWithFrame:frame isPreview:isPreview];
    if (self) {
        [self setAnimationTimeInterval:1.0 / 30.0]; // 30 FPS
        ballRadius = 20.0; // Set ball size
        ballPosition = NSMakePoint(NSWidth(frame) / 2, NSHeight(frame) / 2); // Start in the
        ballVelocity = NSMakeSize(4.0, 4.0); // Set initial velocity
    }
    return self;
}

- (void)startAnimation
{
    [super startAnimation];
}

- (void)stopAnimation
```

```
{
    [super stopAnimation];
}

- (void)drawRect:(NSRect)rect
{
    [super drawRect:rect];

    // Clear the screen with a black background
    [[NSColor blackColor] set];
    NSRectFill(rect);

    // Draw the ball
    [[NSColor redColor] set];
    NSRect ballRect = NSMakeRect(ballPosition.x - ballRadius, ballPosition.y - ballRadius, ba
    NSBezierPath *ballPath = [NSBezierPath bezierPathWithOvalInRect:ballRect];
    [ballPath fill];
}

- (void)animateOneFrame
{
    // Update ball position
    ballPosition.x += ballVelocity.width;
    ballPosition.y += ballVelocity.height;

    // Bounce off walls
    if (ballPosition.x - ballRadius <= 0 || ballPosition.x + ballRadius >= NSWidth(self.bound
        ballVelocity.width = -ballVelocity.width;
    }
    if (ballPosition.y - ballRadius <= 0 || ballPosition.y + ballRadius >= NSHeight(self.bou
        ballVelocity.height = -ballVelocity.height;
    }

    // Request to redraw
    [self setNeedsDisplay:YES];
}

- (BOOL)hasConfigureSheet
{
    return NO; // No configuration sheet for this example
}
```

```
- (NSWindow *)configureSheet
{
    return nil;
}

@end
```

7 Zakljucek

Ohranjevalniki zaslona kljub evoluciji zaslonske tehnologije ostajajo pomemben raziskovalni in razvojni izziv. Ponujajo vpogled v grafične tehnologije, optimizacijo zmogljivosti in ustvarjalne pristope pri oblikovanju interaktivnih uporabniških izkušenj. Čeprav se ne uporabljajo več v svoj prvotem namen so ostali v uporabi zaradi varnostnih in estetskih razlogov. Prav tako so pomembni z vidika varčevanja energije in optimalne uporabe sredstev, ki nam jih ponuja programska oprema računalnikov. Novi izzivi ohranjevalnikov zaslona so projekti prostovoljnega računalništva(angl. volunteer computing), ki omogočajo uporabo domačih računalnikov, za raziskovalne namene.

8 Viri

- https://learn.microsoft.com/en-us/windows/win32/lwef/screen-saver-library
- https://en.wikipedia.org/wiki/Screensaver
- https://developer.apple.com/documentation/screensaver
- https://en.wikipedia.org/wiki/XScreenSaver#cite_note-13
- https://linux.die.net/man/1/xscreensaver
- https://computer.howstuffworks.com/screensaver.htm