آموزش
نکات
برنامه‌نویسی
سیمبین

series60.blogfa.com

مقاله شماره ۴: نحوه استفاده از Time و Timer

از سری مقاله‌های «آموزش نکات برنامه‌نویسی سیمبین»

ویرایش ۱ – ۵ فروردین ۱۳۸۷

سیستم عامل سیمبین

موسی مرادی

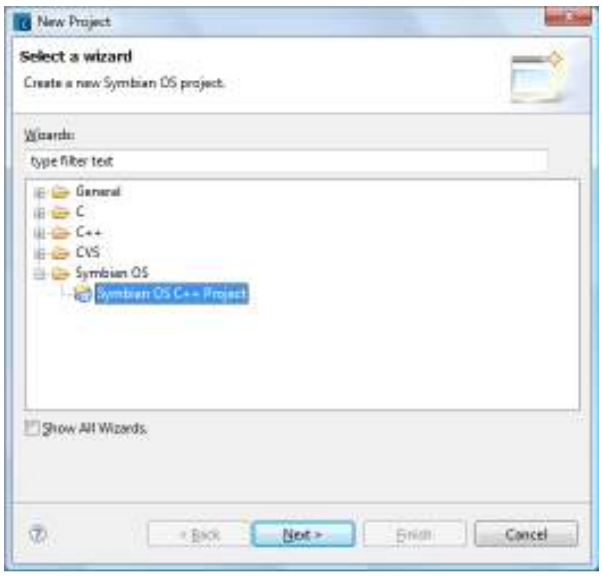
mousamk@gmail.com

این تیوتوریال به شما می‌آموزد که چگونه از کلاس TTime استفاده کنید. همچنین نحوه استفاده از تایمر را هم یاد می‌گیرید. در این تیوتوریال از نرم‌افزار Carbide.C++ استفاده خواهیم کرد. بنابراین برای دنبال کردن این تیوتوریال باید این نرم‌افزار و همچنین یک SDK سازگار با آن را داشته باشید.

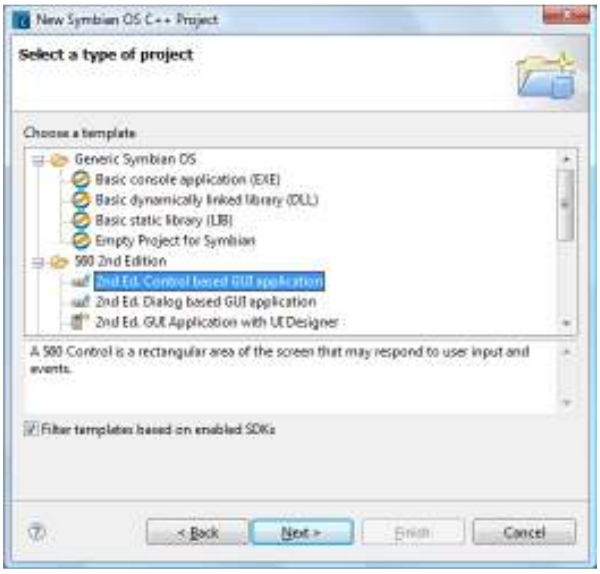
در این تیوتوریال، قصد داریم برنامه‌ای بنویسیم که زمان را در صفحه نمایش دهد.

ساخت پروژه جدید

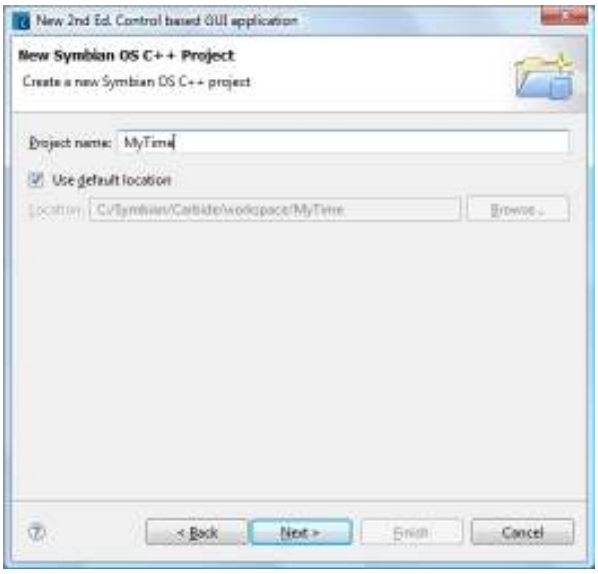
نرم افزار Carbide.C++ را باز کنید و سپس از طریق File -> New -> Project یک پروژه جدید بسازید. طبق تصاویر زیر، ویزارد پروژه جدید را پیش بروید. نام برنامه را هم MyTime بگذارید.



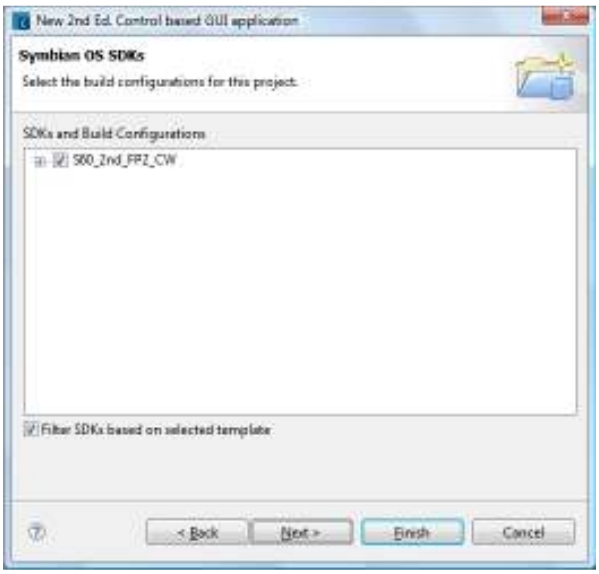
تصویر ۱



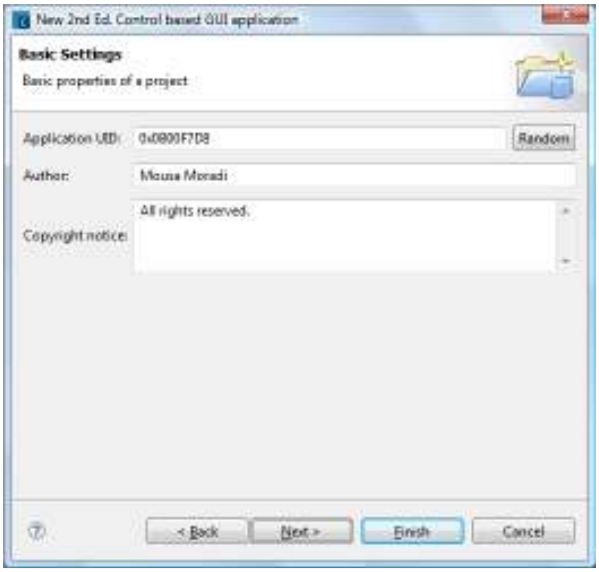
تصویر ۲: انتخاب نوع پروژه (از نوع Control Based)



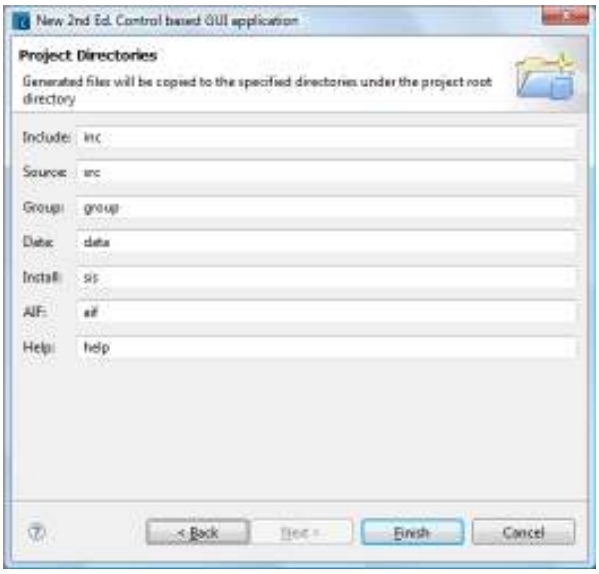
تصویر ۳: نام و مکان ذخیره پروژه



تصویر ۴: انتخاب SDK



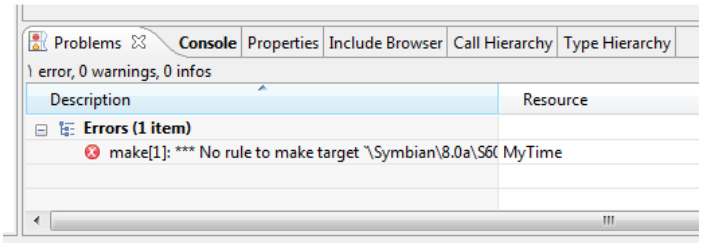
تصویر ۵: UID برنامه و نام نویسنده برنامه و ©



تصویر ۶: فولدرهای ذخیره فایل‌های پروژه

رفع خطا

پس از ساخت پروژه جدید، قبل از انجام هر تغییری، آن را اجرا کنید. اگر همه چیز به خوبی پیش برود، شبیه‌ساز سیمبین را خواهید دید و همچنین برنامه MyTime را که در انتهای منوی شبیه‌ساز است. اما ممکن است به خاطر مشکلات عدم سازگاری کارباید با نرم‌افزارها و ویندوزهای جدید، در اول کار با خطایی شبیه تصویر ۷ روبرو شوید:



تصویر ۷

البته همانطور که در تصاویر قبلی هم دیدید، من از ویندوز ویستا و همچنین از ویژوال استودیو ۲۰۰۵ استفاده می‌کنم، اگر شما از ویندوز xp و از ویژوال استودیو ۲۰۰۳ و یا قبل‌تر، استفاده کنید، احتمالاً این خطا را نداشته باشید.

در صورت بروز این خطا، کارهای زیر را انجام دهید:
فایل bld.inf را باز کنید و خطی که با gnumakefile شروع می‌شود را کامنت کنید. حالا پروژه را دوباره کامپایل کنید و این دفعه دو خطا خواهید گرفت هر دو سطری که این خطاها در آن هستند را کامنت کنید.
با انجام این کار، خطاها رفع می شوند اما منوی Help برنامه MyTime از کار می‌افتد. همچنین در انتها اگر خواستید برنامه را برای گوشی موبایل واقعی کامپایل کنید، باید خط مربوط به فایل Help در فایل pkg را هم کامنت کنید.

حذف کنترل‌های اولیه

اگر برنامه را اجرا کنید، می‌بینید که در صفحه، دو متن نشان داده می‌شود. باید آنها را حذف کنیم. برای این منظور، کارهای زیر را انجام دهید:

در تابع ConstructL از کلاس Container، ۶ خطی که مربوط به ساختن دو label هستند را حذف کنید. همچنین در مخرب (Destructor) کلاس هم دو دستور مربوط به حذف آنها را حذف کنید. همین کار را برای SizeChanged هم انجام دهید. (دو خط را حذف کنید.)

در تابع CountComponentControl به جای ۲، مقدار صفر برگردانید. در تابع ComponentControl هم در ساختار switch، هر دو case موجود را حذف کنید. (فقط default باقی می‌ماند.) و بالاخره در فایل MyTimeContainer.h، دو خطی که مربوط به تعریف دو label هستند را حذف کنید.

اکنون هر دو label از برنامه حذف شده‌اند.

نوشتن کدها

الان به هدف اصلی خودمان برمی‌گردیم. می‌توانیم ساعت را مثل دو متنی که در بالا حذف کردیم، در یک label نشان دهیم و یا اینکه مستقیماً در صفحه بکشیم. من راه دوم را استفاده می‌کنم.

در تابع MyTimeContainer.h در انتهای تعریف کلاس، دو خط زیر را اضافه کنید:

```
TTime iTime;
CPeriodic* iPeriodicTimer;
```

در اینجا یک آبجکت از نوع TTime و یک اشاره‌گر از نوع CPeriodic تعریف کردیم. کلاس TTime برای کار کردن با زمان استفاده می‌شود و حاوی توابع بسیار مفید و کاربردی برای کار کردن با زمان است. از کلاس CPeriodic هم برای ساخت تایمر استفاده می‌شود.

همچنین در ادامه تعاریف بالا، تعریف توابع زیر را بنویسید:

```
void StartTimerL();
static TInt TimerCallBack(TAny* aObject);
void StopTimer();
```

تابع اولی برای ساختن و به کار انداختن تایمر است. دومی هم تابعی است که تایمر در هر دفعه آن را اجرا خواهد کرد. (این تابع باید حتماً static باشد و همچنین مقدار بازگشتی‌اش هم باید از نوع Tint باشد. یک اشاره‌گر هم از نوع نامعلوم می‌گیرد.) تابع سومی هم همان‌طور که از نامش معلوم است، برای متوقف کردن تایمر استفاده می‌شود.

حالا در فایل MyTimeContainer.cpp تابع اولی را به صورت زیر پیاده کنید:

```
void CMyTimeContainer::StartTimerL()
        {
        iPeriodicTimer = CPeriodic::NewL(CActive::EPriorityLow);
        iPeriodicTimer->Start(0, 1000000, TCallBack(TimerCallBack, this));
        }
```

در سطر اول، اشاره‌گر تایمر را new میکنیم و در سطر دوم هم آن را به کار می‌اندازیم. پارامترهایی که تابع Start می‌گیرد، به این صورت هستند:

۱) تأخیر شروع به کار کردن تایمر بعد از ساخته شدن. در اینجا ما صفر داده‌ایم تا بلافاصله اجرا شود.
۲) Interval تایمر بر حسب میکروثانیه. (Interval فاصله بین اجراهای متوالی است.) در اینجا آن را به ۱ ثانیه تنظیم کرده‌ایم، زیرا در هر ثانیه زمان را آپدیت خواهیم کرد.
۳) یک آبجکت از نوع TCallBack که دو پارامتر می‌گیرد. اولی آدرس تابع استاتیکی است که باید در هر دوره، توسط تایمر صدا زده شود. دومی هم اشاره‌گری است که به تابع استاتیک پاس خواهد شد.

تابع TimerCallBack را هم به این صورت بنویسید:

```
TInt CMyTimeContainer::TimerCallBack(TAny* aObject)
      {
      (static_cast<CMyTimeContainer*>(aObject))->iTime.UniversalTime();
      (static_cast<CMyTimeContainer*>(aObject))->DrawDeferred();

      return 1;
      }
```

در اینجا ابتدا اشاره‌گر داده شده را به نوع کلاس Containerمان، تبدیل (cast) می‌کنیم و سپس روی آبجکت iTime از کلاس Container، تابع UniversalTime را صدا می‌زنیم. این تابع، زمان فعلی (جهانی) را در آبجکت از نوع TTime، ذخیره می‌کند. (می‌توانید از تابع HomeTime هم استفاده کنید. این تابع هم زمان محلی را در آبجکت ذخیره می‌کند.) سپس در سطر بعد تابع DrawDeferred را روی همان اشاره‌گر کلاس Container صدا می‌زنیم. این تابع به سیستم می‌گوید که اطلاعاتی که در صفحه باید رسم شوند، تغییر کرده‌اند و باید دوباره رسم شوند. یعنی در حقیقت تابع Draw باید صدا زده شود. توجه داشته باشید که سیستم دقیقاً بعد از این تابع، تابع Draw را اجرا نمی‌کند، بلکه طبق اولویت‌بندی پروسس‌ها، هر موقع که از لحاظ اولویت، توانست، این کار را انجام می‌دهد. اگر می‌خواهید که دقیقاً همین الان رسم شود، به جای DrawDeferred، تابع DrawNow را صدا بزنید.

معمولاً رسم بر این است که این تابع استاتیک، که توسط تایمر اجرا می‌شود، یک تابع غیراستاتیک از کلاس اشاره‌گر پاس شده به خودش را صدا می‌زند و تمام کارها در آن انجام می‌شوند، ولی در اینجا چون کار زیادی برای انجام نداریم، در همین تابع استاتیک، کارمان را انجام می‌دهیم.

و بالاخره تابع StopTimer را هم به این صورت پیاده‌سازی کنید:

```
void CMyTimeContainer::StopTimer()
      {
      if(iPeriodicTimer)
            {
            iPeriodicTimer->Cancel();
            delete iPeriodicTimer;
            iPeriodicTimer = NULL;
            }
      }
```

در این تابع، ابتدا بررسی می‌کنیم که اگر تایمر وجود دارد، آن را متوقف کرده و سپس حذف می‌کنیم. از این تابع، دو نکته مهم را در برنامه‌نویسی سیمبین، یاد می‌گیریم:

- در مخرب‌های کلاس‌ها، همیشه با این فرض که آبجکت‌های کلاس ممکن است وجود نداشته باشند، آنها را بررسی و حذف کنیم. زیرا ممکن است موقع ساخت آن آبجکت‌ها (در تابع ConstructL)، leave رخ دهد و بعضی از آبجکت‌های کلاس ساخته نشوند.
- در برنامه‌نویسی سیمبین، هر موقع که می‌خواهیم اشاره‌گری را delete کنیم، بلافاصله بعد از حذف کردن آن، آن را صفر (NULL) می‌کنیم. علت این امر هم این است که بعداً بتوانیم مورد بالا را در مورد آن اشاره‌گر اجرا کنیم. زیرا اگر اشاره‌گر را صفر نکنیم، اشاره‌گر هنوز هم به آن محلی از حافظه (که الان دیگر مدیریتش در دست ما نیست) اشاره می‌کند و بعداً ممکن است دچار دو بار delete شدن شود. البته این نکته را فقط در جایی استفاده می‌کنیم که دوباره می‌خواهیم آن اشاره‌گر را به حافظه جدیدی نسبت دهیم و بنابراین در مخرب‌ها رعایت این امر لازم نیست و من در اینجا فقط برای توضیح، این کار را کرده‌ام.

معمولاً ما یک ماکرو تعریف می‌کنیم (مثلاً با نام DELETE_AND_NULL) که این دو کار را با هم انجام دهد و دیگر در همه جا، دو دستور را ننویسیم. ولی در اینجا چون فقط یک بار این عمل را انجام می‌دهیم، نیازی به تعریف ماکرو نیست.

استفاده از توابع نوشته شده

حالا تمام توابع لازم برای کار کردن آماده‌اند. تابع StartTimerL را در انتهای ConstructL از کلاس CMyTimeContainer و تابع StopTimer را هم در مخرب این کلاس، صدا بزنید. در این صورت، همین که برنامه‌مان در موبایل (و یا شبیه‌ساز) اجرا شد، تایمر به کار می‌افتد و موقع بستن برنامه هم تایمر متوقف می‌شود.

الان برنامه ما به درستی کار می‌کند، اما اگر برنامه را اجرا کنید، چیزی نخواهید دید، که البته کاملاً بدیهی است، زیرا ما چیزی را نمایش نداده‌ایم. برای دیدن نتیجه، باید ساعت را در صفحه رسم کنیم. برای این کار، به انتهای تابع Draw در کلاس CMyTimeContainer، کدهای زیر را اضافه کنید:

```
TBuf<10> buf;
iTime.FormatL(buf, _L("%H:%T:%S"));
gc.SetPenColor(KRgbBlack);
gc.UseFont(CEikonEnv::Static()->TitleFont());
gc.DrawText(buf, TRect(30, 30, 146, 52), 20, CGraphicsContext::ECenter, 0);
gc.DiscardFont();
```

در اینجا ابتدا یک واصف (descriptor = رشته‌ها در سیمبین) به طول حداکثر ۱۰ تعریف می‌کنیم. حالا می‌خواهیم یک نمایش تصویری از ساعت داشته باشیم. برای این منظور، روی آبجکت iTime، تابع FormatL را صدا می‌زنیم و دو واصف به آن می‌فرستیم. دومی نوع نمایش ساعت را مشخص می‌کند (%H با ساعت، %T با دقیقه و %S هم با ثانیه جایگزین خواهد شد. برای اطلاعات بیشتر در مورد تابع FormatL به مستندات SDK مراجعه کنید.) پارامتر اول هم مرجع (reference)ی است که متن فرمت شده نمایش ساعت، در آن قرار خواهد گرفت.

حالا نمایش متنی ساعت را در دست داریم. اکنون ابتدا رنگ قلم را به سیاه تنظیم می‌کنیم. سپس فونت Title را برای نوشتن انتخاب می‌کنیم. (بقیه فونت‌های قابل استفاده را می‌توانید در مستندات SDK در توضیحات کلاس CEikonEnv مشاهده کنید.) و سپس توسط تابع DrawText، متن را در صفحه رسم می‌کنیم. این تابع دارای Overloadهای مختلفی است و تابعی که ما در اینجا استفاده کرده‌ایم، پارامترهایی به این شرح می‌گیرد:

۱) متنی که باید نمایش داده شود.

۲) مستطیلی که متن داخل آن کشیده می‌شود.

۳) خط زمینه داخل مستطیل. در اینجا ما ۲۰ داده‌ایم به این معنی که خط زمینه‌مان دارای عمق ۲۰ پیکسل از بالای مستطیل است.

۴) ترازبندی نمایش متن. در اینجا ما متن را در وسط مستطیل رسم کرده‌ایم. حالت‌های دیگر، ELeft و ERight هستند.

۵) پارامتر آخری هم حاشیه از چپ (برای حالتی که پارامتر قبلی چپ‌چین باشد) و یا راست (برای حالتی که پارامتر قبلی راست‌چین باشد) است. در اینجا ما صفر داده‌ایم. البته مقدار پیش‌فرض این پارامتر هم صفر است. یعنی کلاً می‌توانستیم این پارامتر را به تابع پاس نکنیم.

پیشنهاد می‌کنم پارامترهای این تابع را تغییر دهید و نتیجه‌های مختلف را ببینید.

* ضمناً در اینجا از کلاسی با نام CEikonEnv استفاده کرده‌ایم. پس باید تابع eikenv.h را ضمیمه کنید.

تغییر رنگ

در همین تابع Draw، در کدهای اولیه خطی به صورت زیر وجود دارد:

```
gc.SetBrushColor( KRgbGray );
```

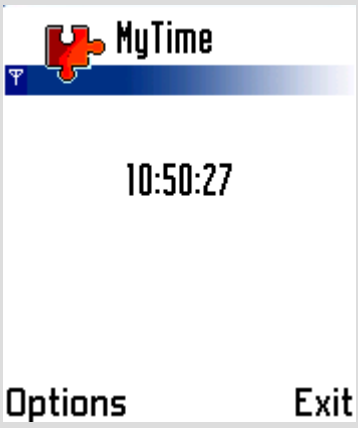
من آن را به این صورت تغییر دادم، تا رنگ پس‌زمینه، سفید شود:

```
gc.SetBrushColor( KRgbWhite );
```

شما هم می‌توانید آن را به رنگ دلخواه تغییر دهید. تعدادی از رنگ‌های معروف مثل سفید و سبز و سیاه و ... به صورت ثوابتی مانند KRgbWhite، KRgbGreen، KRgbBlack و ... تعریف شده‌اند که می‌توانید از آنها استفاده کنید، و برای ساخت رنگ‌های دلخواه هم، می‌توانید از تابع TRgb استفاده کنید. این تابع سه پارامتر می‌گیرد که به ترتیب، مقدار ترکیب رنگ‌های قرمز، سبز و آبی در رنگ نهایی است که هر کدام، اعدادی بین ۰ و ۲۵۵ هستند.

اکنون می‌توانید برنامه را اجرا کنید و از نتیجه اجرای برنامه لذت ببرید!

در تصاویر زیر هم، اجرای برنامه را در شبیه‌ساز و همچنین در موبایل من، مشاهده می‌کنید:



درباره این مقاله:

این مقاله چهارمین مقاله از سری مقاله‌های **آموزش نکات برنامه‌نویسی سیمبین** می‌باشد که در زمینه «نحوه استفاده از Time و Timer» نوشته شده است.

این مقاله که به صورت تیوتوریال نوشته شده است، تألیف خود بنده است.

برای دریافت مقالات دیگر این سری می‌توانید به وبلاگ من (http://series60.blogfa.com) و یا فروم مخصوص برنامه نویسی سیمبین مراجعه کنید.

نویسنده: موسی مرادی کندلجی (mousamk@gmail.com)

در صورتی که هرگونه سؤال، پیشنهاد، انتقادی و یا مطلبی داشتید، می‌توانید از طریق ایمیل تماس بگیرید. ضمناً می‌توانید از طریق وبلاگ و یا فروم هم با من ارتباط داشته باشید.