

DoForMe!

Dokumentacja użytkownika

Jacek Topolski

[1. Wstęp 4](#_Toc340174027)

[2. Skrypty 5](#_Toc340174028)

[2.1. Dodawanie 5](#_Toc340174029)

[2.2. Usuwanie 6](#_Toc340174030)

[2.3. Opóźnienia 6](#_Toc340174031)

[2.4. Klawiatura 7](#_Toc340174032)

[2.4.1. sendText("") 7](#_Toc340174033)

[2.4.2. Print Screen 8](#_Toc340174034)

[2.5. Mysz 8](#_Toc340174035)

[2.5.1. Zmiana współrzędnych 9](#_Toc340174036)

[2.5.2. Przyciski 9](#_Toc340174037)

[2.6. Lista funkcji 10](#_Toc340174038)

[2.7. Stan ekranu 10](#_Toc340174039)

[2.8. Omówienie przykładu 10](#_Toc340174040)

[3. Zarządzanie akcjami 12](#_Toc340174041)

[3.1. Kalendarz 12](#_Toc340174042)

[3.1.1. Legenda 12](#_Toc340174043)

[3.2. Dodawanie 13](#_Toc340174044)

[3.3. Zaznaczanie 14](#_Toc340174045)

[3.4. Edycja & usuwanie 14](#_Toc340174046)

[3.5. Odłączanie 14](#_Toc340174047)

[3.6. Przesuwanie 15](#_Toc340174048)

[3.7. Zapisywanie 15](#_Toc340174049)

[4. Tryb "czuwania" 15](#_Toc340174050)

[4.1. Menu trybu "czuwania" 16](#_Toc340174051)

[4.2. Uruchomienie akcji 16](#_Toc340174052)

[4.3. Powiadomienia 17](#_Toc340174053)

[5. Nagrywanie akcji 17](#_Toc340174054)

[5.1. Uruchomienie 18](#_Toc340174055)

[5.2. Ustawienia 18](#_Toc340174056)

[5.3. Pobieranie stanu ekranu 19](#_Toc340174057)

[5.4. Zatrzymanie 20](#_Toc340174058)

1. Wstęp

Program służy do zautomatyzowania oraz planowania, zdefiniowanych przez użytkownika, zadań "zleconych" dla systemu (tzw. akcje). Umożliwia on m.in. dokładne sterowanie urządzeniami wejścia tj. mysz lub klawiatura. Aby użytkownik zdefiniował zadanie dla systemu musi napisać skrypt, w którym są udostępnione funkcje umożliwiające sterowanie myszką lub klawiaturą. Dzięki temu, że program posiada wbudowany silnik dla języka skryptowego LUA, możliwe jest wykorzystanie potencjału jaki drzemie w tymże języku.

Ponadto, oprócz samego definiowania zadań, możliwe jest także zarządzanie nimi poprzez określenie konkretnego dnia i godziny, w którym dane zadanie ma się uruchomić. Dzięki temu program po określeniu szeregu zadań nie wymusza ponownej interakcji z użytkownikiem w celu wywołania danego zadania. Służy do tego wbudowany kalendarz, na który umieszczamy napisane przez nas skrypty. Oprócz określenia dnia i godziny, możemy także ustawiać powtórzenia skryptu, czyli np. aby nasze zadanie było wykonywane co 2 dni oraz zawsze w piątki z wyłączeniem określonych przez nas dni.

Istnieje także funkcjonalność automatycznego nagrywania skryptu na podstawie działań użytkownika w systemie. Nagrywanie obejmuje oba urządzenia wejściowe, tj. mysz oraz klawiaturę.

Program do symulacji urządzeń wejścia korzysta ze specyficznych części systemu Windows dlatego też aplikacja działa tylko na tym systemie.

1. Skrypty

Na podstawie przykładowego zadania będą omówione poszczególne części programu.

* 1. Dodawanie

Na początek tworzymy skrypt wybierając Script->New... i wpisując nazwę dla naszego skryptu. Należy pamiętać, że wpisując nazwę jest to tak naprawdę nazwa dla pliku, w którym owy skrypt będzie rezydować, zatem musimy przestrzegać zasad nazewnictwa plików[[1]](#footnote-2) w tym samym folderze[[2]](#footnote-3) (mówiąc to mam na myśli, że nazwy muszą być unikalne). Następnie naciskamy przycisk "Ok" i jesteśmy gotowi do pisania skryptu lub nagrywania. Należy pamiętać, że aby nagrać nowy skrypt musimy najpierw utworzyć na niego miejsce, ponieważ istnieje możliwość "dogrania" skryptu - takie miejsce możemy utworzyć poprzez utworzenie całkowicie nowego skryptu lub wybranie istniejące z listy.

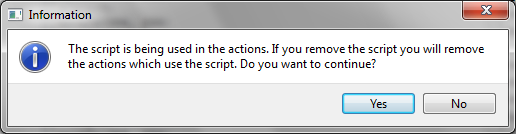
Załóżmy, że naszym celem będzie narysowanie okręgu w programie Paint. Naszym skryptem może być w takim razie na przykład:

1. sendText("{rwin}")
2. sendText("paint{n}")
3. sendText("{lalt+} {lalt-}")
4. sendText("s")
5. sendText("{lctrl+}w{lctrl-}")
6. sendText("a")
7. sendText("{t}{t}{t}{t}{t}")
8. sendText("{right}{t}160{t}100")
9. sendText("{n}")
10. sendText("{lalt}ssh{right}{right}")
11. sendText("{n}")
12. setInterval(20)
13. sendText("{lshift+}")
14. leftDownAt(50, 150)
15. leftUpAt(130, 230)
16. sendText("{lshift-}")

Zauważmy, że do wybierania jakiejś opcji czy narzędzia generalnie korzystamy z klawiatury. Jest to zrobione w taki sposób z tego powodu, że każdy użytkownik może mieć ustawioną różną rozdzielczość ekranu oraz różnie ustawione/przeskalowane okno aplikacji (dopóki nie wykonamy trzeciej komendy - maksymalizacja okna), zatem jawne użycie współrzędnych w funkcjach dla myszki mogło by nie zadziałać w niektórych przypadkach.

* 1. Usuwanie

Usuwanie skryptu wykonujemy poprzez kliknięcie przycisku Remove w prawym górnym rogu listy ze skryptami. Należy mieć na uwadze to, że jeśli będziemy chcieli usunąć skrypt, który jest dodany jako zadanie na kalendarzu, zostaniemy o tym poinformowanie i zmuszeni do potwierdzenia. Gdy usuniemy skrypt - zadania także zostaną usunięte.



* 1. Opóźnienia

Zanim jednak dojdziemy do omawiania jakiegokolwiek przykładu, trzeba powiedzieć parę słów na temat ogólnego działania skryptów. Po pierwsze każda linia kodu jest traktowana jako oddzielna komenda. Jest to istotne w momencie ustawiania opcji "Starting delay" w opcjach odtwarzacza skryptów "Player settings" (F6). Dla przykładu gdy ta wartość będzie równa 5000 (milisekund) to dla takiego skryptu:

1. sendText("a")
2. sendText("{n}")

po kliknięciu Play (zielona strzałka) najpierw odczekamy 5 sekund, po czym zostanie wciśnięte "a" na klawiaturze, następnie znowu musi minąć 5 sekund aby został wciśnięty Enter. Istotne jest także wykorzystywanie "scalania" funkcji sendText(), tzn. dla takiego skryptu:

1. sendText("a{n}")

najpierw odczekamy 5 sekund, po czym zostanie wciśnięte "a" a następnie (bez 5 sekundowej przerwy) Enter.

Pozostając przy temacie opóźnień, do dyspozycji mamy także funkcję sleep(ms), która dodaje opóźnienie pomiędzy komendami. Na przykład (dla 5000 "Starting delay"):

1. sendText("a")
2. sleep(1000)
3. sendText("{n}")

najpierw odczekamy 5 sekund, następnie zostanie wypisanie "a" oraz po 6 sekundach (5000ms ze "Starting delay" + 1000ms ze sleep'a) zostanie wciśnięty Enter.

Istnieje także metoda setInterval(ms), która oznacza, że od tego momentu wartość "Starting delay" przyjmuje wartość podaną jako argument dla setInterval(ms).

* 1. Klawiatura

Symulacja klawiatury składa się z realnych czynności wykonywanych na klawiaturze, mamy tu na myśli to, że np. przy napisaniu symbolu "ą" trzeba nacisnąć prawy ALT, następnie klawisz A i zwolnić ALT.

* + 1. sendText("")

Skupmy się na razie na funkcji sendText(), gdyż jest to chyba najbardziej skomplikowana funkcja. Funkcja ta przyjmuje jako argument ciąg znaków, które zostaną wciśnięte na klawiaturze. W przypadku prostych liter typu "A", "a" id. wystarczy je wpisać explicite, tzn. sendText("Aa"). Jednak istnieją także znaki specjalne, typu "Enter", "Escape", "Shift" itd. Znaki specjalne umieszczane są pomiędzy { a } symbolem, np. {n} symbolizuje kliknięcie Enter'a, a {t} tabulatora. Znaki te możemy znaleźć po prawej stronie naszego pola tekstowego w zakładce Keyboard.

Należy pamiętam, że takie wpisanie {n} oznacza wciśnięcie oraz natychmiastowe puszczenie przycisku. Istnieją jednak znaki, które muszą być przytrzymane na dłużej w czasie gdy inne klawisze są wciskane, są to: Shift, Control oraz Alt. Aby uporać się z tego typu problemami możemy użyć jednego z trzech stanów określających stan klawisza, tzn. wciśnięcie, zwolnienie oraz kliknięcie (czyli wciśnięcie a następnie zwolnienie). Znak oznaczony jako wciśnięcie wpisujemy jako {ralt+}, zwolnienie {ralt-} oraz kliknięcie {ralt}, zatem gdy chcemy napisać literę "ą" używamy:

1. sendText("{ralt+}a{ralt-}")

musimy pamiętać o zwolnieniu klawisza, bo jak się tego nie zrobi mogą się dziać dziwne rzeczy z klawiaturą[[3]](#footnote-4). Zauważmy, że nie musimy stosować wciśnięcia "Shift'a" w celu napsiania wielkiej litery, program sam to będzie potrafił przerobić. Jednak polskie znaki nie są przerabiane, ponieważ nie należą one do tablicy ASCII.

* + 1. Print Screen

Istotny jest także klawisz "Print, Screen", który posiada tak naprawdę dwa znaczenia, zatem gdy użytkownik chce pobrać zrzut ekranu, powinien użyć symbolu {ss} zamiast {print}.

* 1. Mysz

W przypadku myszy komputerowej sprawa jest znacznie prostsza (jest dużo więcej funkcji, lecz są one trywialne). Mamy do dyspozycji trzy przyciski (dla każdego 3 stany) oraz przesuwanie kursora w wybrane przez nas współrzędne. W dzisiejszych czasach myszki mają oczywiście znacznie więcej przycisków, jednak nie jest to w żaden sposób ustandaryzowane jak w przypadku klawiatury np. QWERTY, aczkolwiek nawet tutaj pewne klawisze mogą się różnić.

Należy mieć na uwadze to, że gdy chcemy stworzyć uniwersalny skrypt, to zazwyczaj nie powinniśmy korzystać z myszki z uwagi na to, że użytkownik może mieć na przykład inną rozdzielczość lub inaczej rozłożone elementy w systemie.

* + 1. Zmiana współrzędnych

Do zmiany współrzędnych myszy mamy do dyspozycji funkcję moveTo(x, y), która przesuwa nasz kursor na pozycję (x, y).

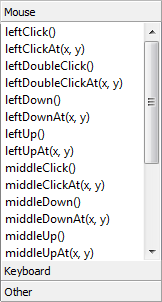
* + 1. Przyciski

Użycie przycisków jest równie proste jak i przesuwanie kursora. Do dyspozycji mamy 10 zwykłych funkcji (trzy przyciski, w których każdy może mieć trzy stany oraz jedna funkcja odpowiadająca za podwójne lewe kliknięcie - leftDoubleClick()). Mówiąc "zwykłych" mamy na myśli te funkcje, które nie korzystają z dodatkowych argumentów określających gdzie ma zostać przyciśnięty przycisk. Innymi słowy dla przykładu leftClick() kliknie lewym przyciskiem w miejscu, w którym aktualnie znajduje się kursor.

Aby kliknąć we wskazane miejsce możemy albo użyć jawnie obu funkcji, tzn. przesunięcie kursora a następnie kliknięcie przycisku lub użyć jednej ze skróconych form z sufiksem "At"[[4]](#footnote-5), które przyjmują jako argumenty dwie liczby określające współrzędne na ekranie. Istotnym faktem jest tutaj to, że mimo, że widzimy to jako jedną komendę (patrz dział "2.1 Opóźnienia"), to tak naprawdę są to dwie oddzielnie komendy, zatem:

1. leftClickAt(x, y)

jest równoważny z kodem:

1. moveTo(x, y)
2. leftClick()
   1. Lista funkcji

Pełny wgląd w udostępnione funkcje mamy po prawej stronie pola tekstowego, w którym piszemy nasz skrypt. W zakładce "Keyboard" oprócz funkcji sendText("") zostały wstawione także symbole oznaczające znaki specjalne. Aby samemu nie wpisywać tych funkcji wystarczy podwójne kliknięcie na odpowiednią pozycję na liście, aby wstawić komendę do pola tekstowego w miejscu, w którym aktualnie znajduje się kursor.

* 1. Stan ekranu

Istnieje także funkcja waitForImage(path\_to\_bmp)przyjmująca jako argument ścieżkę do pliku \*.bmp, umożliwiająca sprawdzenie czy dany fragment obrazka jest wyświetlony na ekranie. Jeśli nie jest, skrypt w tym miejscu zostanie wstrzymany i będzie kontynuował dopiero gdy się wyświetli. Dzięki tej funkcjonalności można w dosyć uniwersalny sposób sprawdzać czy np. jakaś akcja w systemie już się zakończyła. Ponadto gdy skrypt się "zgubi" możemy nakierować go na poprawną ścieżkę wykonując odpowiednie działania i gdy się "odnajdzie" pozwolić mu dalej kontynuować. Funkcjonalność ta zostanie także opisania w momencie omawiania Record'a.

* 1. Omówienie przykładu

Powróćmy w końcu do naszego pierwotnego przykładu, czyli rysowanie okręgu w programie Paint. Jeśli przeczytałeś poprzednie rozdziały to praktycznie wiesz już wszystko na temat tego kodu, jeśli nie to teraz pokrótce omówimy każdą linijkę z osobna.

Na samym początku uruchamiamy program poprzez wyświetlenie menu start (zazwyczaj klawisz jest oznaczony jako flaga z logiem Windows), wpisanie słowa "paint" oraz kliknięcie enter:

1. sendText("{rwin}")
2. sendText("paint{n}")

następnie ustawiamy aplikację jako zmaksymalizowaną (zauważmy, że pomiędzy przyciśnięciem a zwolnieniem lewego ALT jest naciśnięcie spacji! Spacja może być także wywołania poprzez {space}):

1. sendText("{lalt+} {lalt-}")
2. sendText("s")

następnym krokiem jest ustawienie rozmiaru obszaru do rysowania, używając skrótów programu Paint wchodzimy do odpowiedniego okna dialogowego i zmieniamy wartości po czym wciskamy Enter aby zatwierdzić.

1. sendText("{lctrl+}w{lctrl-}")
2. sendText("a")
3. sendText("{t}{t}{t}{t}{t}")
4. sendText("{right}{t}160{t}100")
5. sendText("{n}")

kolejnym krokiem jest wybranie narzędzia do rysowania, w naszym przypadku jest to narzędzie do rysowania okręgu.

1. sendText("{lalt}ssh{right}{right}")
2. sendText("{n}")

w tym miejscu ustawiamy opóźnienie na 20 milisekund, aby rysowanie wykonywało się szybciej. Wcześniej warto było ustawić większe opóźnienie z tego względu, że komputery mogą różnić się szybkością, zatem więcej czasu im może być potrzebne do np. uruchomienia programu[[5]](#footnote-6).

1. setInterval(20)

ostatnim krokiem jest narysowanie okręgu z przytrzymanym klawiszem Shift, dzięki temu otrzymujemy okrąg a nie elipsę.

1. sendText("{lshift+}")
2. leftDownAt(50, 150)
3. leftUpAt(130, 230)
4. sendText("{lshift-}")

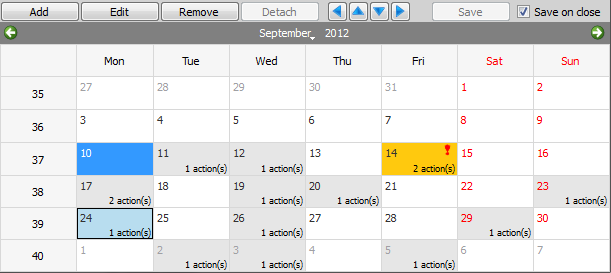
Teraz wystarczy kliknąć w zieloną strzałkę aby zobaczyć efekty (lub naciskając F5). Jeśli jakiś długi skrypt został uruchomiony (np. przez przypadek) możemy go przerwać przytrzymując kombinację klawiszy lewy ctrl + lewy shift + X do momentu gdy wyświetlony zostanie komunikat.

Po zakończeniu wykonywania skryptu, system poinformuje nas o tym wyświetlając stosowny komunikat, który po 3 sekundach automatycznie zniknie (lub po kliknięciu przycisku Ok).

1. Zarządzanie akcjami

Po napisaniu skryptu chcielibyśmy aby został on wykonany w określonym dniu o określonej przez nas godzinie. Do tego celu istnieje kalendarz, na który możemy "nakładać" nasze akcje (lub inaczej zadania).

* 1. Kalendarz



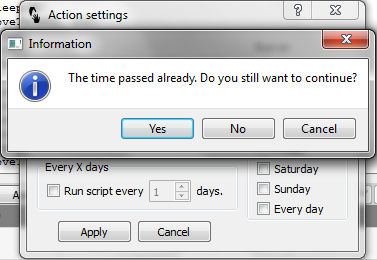
Każdy "stan" komórki jest podkreślony w różny sposób. Mówiąc stan mamy na myśli dzień, który jest np. zaznaczony, posiada jakąś akcję do wykonania czy konflikt do wyeliminowania.

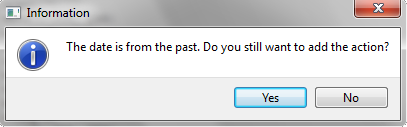
* + 1. Legenda

|  |  |
| --- | --- |
|  | Aktualna data |
|  | Aktualnie zaznaczona akcja |
|  | Dni, w których występuje co najmniej 1 akcja |
|  | Dni, w których istnieje konflikt (dwie akcje o tej samej godzinie) |
|  | Aktualnie zaznaczona data |
| exclamation.png | Dni, w których istnieje konflikt (dwie akcje o tej samej godzinie) |

Zauważmy, że konflikt akcji jest oznaczony na dwa sposoby. Jest tak z tego względu, że może się zdarzyć, że jedna z akcji w tym dniu zostanie zaznaczona - czyli zmieni kolor na jasno niebieski, dlatego niezbędne jest oznaczenie konfliktu dodatkowym symbolem, aby było to widoczne. Możliwe było oznaczenie konfliktu tylko po prostu wykrzyknikiem, jednak wtedy byłby on o wiele mniej widoczny.

* 1. Dodawanie

 Aby dodać nasz uprzednio stworzony skrypt musimy wybrać na kalendarzu odpowiednią datę i kliknąć przycisk "Add", który wyświetli okno dialogowe (lub komunikat, że ten dzień już minął) z opcjami umożliwiającymi określenie godziny, powtórzeń oraz skryptu, który ma się uruchomić. Następnie w celu potwierdzenia klikamy przycisk "Apply". Jednak jeśli próbujemy dodać akcję dla dnia (lub godziny jeśli jest to aktualny dzień), który już minął, to zostaniemy o tym poinformowani odpowiednim komunikatem.



Co więcej podczas dodawania akcji do kalendarza sprawdzana jest jego poprawność, dzięki temu nie jest możliwe dodanie akcji, które zwrócą błąd przy uruchomieniu.

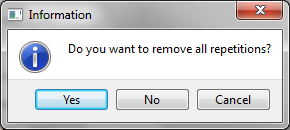
* 1. Zaznaczanie

Jeśli w danym dniu jest tylko jedna akcja do wykonania, to zostanie ona automatycznie zaznaczona po wybraniu dnia na kalendarzu bez konieczności zaznaczania jej na liście akcji (lista po lewej stronie kalendarza), w innym przypadku wymagane jest sprecyzowanie, którą akcję chcemy wybrać.

* 1. Edycja & usuwanie

Aby uaktywnić przyciski do edycji oraz usuwania akcji (przyciski nad kalendarzem), należy uprzednio wybrać jakąś akcję z kalendarza.

Edycja umożliwia nam powtórne ustawienie opcji (powtórzeń, godziny i skryptu) dla akcji, tak jak przy jej dodawaniu. Należy mieć na uwadze to, że ponowna edycja akcji z powtórzeniami nie przywróci usuniętych pojedynczych akcji.

 Działanie usuwania jest zależne od akcji którą wybraliśmy. Jeśli jest to akcja bez powtórzeń, to zostanie ona po prostu usunięta. Jeśli jednak akcja posiada powtórzenia, to zostaniemy zapytani czy chcemy usunąć wszystkie akcje powiązane z zaznaczoną akcją czy tylko aktualnie wybraną.

* 1. Odłączanie

Załóżmy, że chcemy teraz stworzyć akcję z powtórzeniami, która wykonuje się od dnia 10.09 co dwa dni. Zatem akcja ta zostanie wywołana w dniach: 10.09, 12.09. 14.09 itd. Załóżmy teraz, że wszystko jest tak jak chcemy oprócz jednej rzeczy - chcielibyśmy aby akcja z dnia 12.09 przeniosła się na 13.09. Możemy to osiągnąć na dwa sposoby. Pierwszy polega po prostu na usunięciu akcji z dnia 12.09 i dodaniu jej na nowo (bez powtórzeń) do dnia 13.09. Drugim sposobem jest tytułowe odłączanie. Służy do tego przycisk "Detach", który jak nazwa wskazuje, odłącza akcję od reszty. Innymi słowy, gdy odłączymy akcję z dnia 12.09, szereg akcji się nie zmieni, tzn. dalej będzie to samo wywołane w tych samych dniach. Jedyną zmianą będzie to, że teraz po wybraniu akcji z dnia 12.09 nie zaznaczymy całej reszty a tylko tą jedną odłączoną, którą możemy bez problemów przesuwać na kalendarzu. W naszym przypadku przesuniemy ją po prostu na dzień 13.09.

* 1. Przesuwanie

move.bmp Jak już zostało wspomniane w poprzednim paragrafie, przyciski odpowiadające za przesuwanie akcji na kalendarzu są oznaczone niebieskimi trójkątami symbolizującymi kierunek przesuwania. Jeśli dana akcja ma powtórzenia, to przesuwanie będzie oddziaływać na tą akcję oraz jej wszystkie powtórzenia.

* 1. Zapisywanie

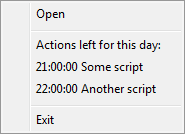
save_on_close.bmp Akcje po dodaniu do kalendarza nie są od razu zapisywane do bazy danych aplikacji. Zapisanie do bazy następuje po kliknięciu przycisku "Save" nad kalendarzem, aczkolwiek domyślnie jest zaznaczone pole "Save on close", które informuje program, że zapisanie do bazy ma następować przy zamknięciu programu[[6]](#footnote-7). Dane zapisywane są do bazy SQLite w pliku actions.

1. Tryb "czuwania"

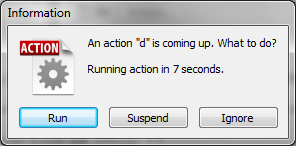
Jeżeli skończyliśmy dodawać akcje na kalendarzu, to chcielibyśmy ustawić aplikację w tryb "czuwania", aby po nastaniu odpowiedniej daty i godziny wywołało ustaloną przez nas akcję. Tak się jednak nie stanie. Trzeba zaznaczyć, że jeśli tylko ustawimy akcje na kalendarzu to akcje nie zostaną wywołane o podanej godzinie.

Tryb "czuwania" włączamy poprzez kliknięcie w to_tray.bmp. Wtedy okno naszej aplikacji ukrywa się i wyświetla się ikona w obszarze ikon systemowych (ikonki w prawym dolnym rogu koło daty[[7]](#footnote-8)). Po przejściu w owy tryb akcje, która muszą zostać wykonane w aktualnym dniu zostają "zakolejkowane", czyli po prostu oczekują na wywołanie o określonej godzinie.

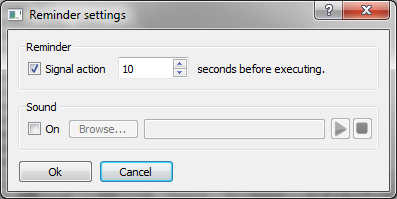
* 1. Menu trybu "czuwania"

 Menu kontekstowe wywołane poprzez prawe kliknięcie myszką na ikonę zawiera listę akcji, które czekają na wywołanie. Jeśli jakaś akcja została wywołana (lub jej nastawiona godzina jest wcześniejsza niż aktualna), to nie zostanie ona wyświetlona na liście. Oprócz listy akcji możliwe jest także powrócenie do poprzedniego trybu zarządzania skryptami i akcjami poprzez kliknięcie pozycji "Open". Możliwe jest także przywrócenie okna poprzez dwukrotne kliknięcie lewym myszki na ikonie. Pozycja "Exit" wyłącza program.

* 1. Uruchomienie akcji

 Gdy nadejdzie w końcu czas określony dla akcji, skrypt zostanie po prostu uruchomiony. Uruchamianie skryptu możemy ustawić tak, że zostaniemy o tym poinformowani chwile przed czasem (domyślne ustawienie 10s). Modyfikacji powiadomień możemy dokonać w opcjach "Reminder settings" w menu Tools lub klikając F8.

* 1. Powiadomienia

 Do wyboru mamy m.in. okienko dialogowe wyświetlone przed uruchomieniem akcji, które posiada trzy opcje - uruchomienie (Run), wstrzymanie (Suspend) oraz zignorowanie akcji (Ignore). Jeśli naszą ustawioną wartością czasową było np. 10 sekund, tzn. że okienko dialogowe będzie wyświetlone przez 10 sekund, jeśli użytkownik nie wybierze żadnej z opcji, okienko zniknie po czym wywołana zostanie akcja.

Dodatkową formą powiadomienia jest sygnał dźwiękowy. Wraz z programem dostępny jest jeden sygnał dźwiękowy, aczkolwiek opcja powiadomienia dźwiękowego jest domyślnie wyłączona. Jeśli domyślny dźwięk nie będzie pasować użytkownikowi, możliwa jest jego zmiana. Program obsługuje popularne formaty takie jak .mp3, .wav, .ogg i wiele innych. Jeśli z jakichś powodów sygnał nie jest odtwarzany przez program jest to wina systemu, ponieważ program korzysta tak naprawdę z systemowych komponentów do odtwarzania dźwięku.

1. Nagrywanie akcji

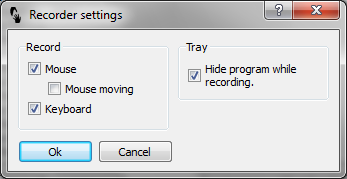
Czasami użytkownik nie ma czasu, ochoty lub, przy pierwszym starciu z aplikacją, dokładnej wiedzy na temat udostępnionych funkcji sterujących urządzeniami. Z pomocą przychodzi funkcjonalność, którą jest tzw. Recorder. Umożliwia on nagrywane działań użytkownika, poprzez automatyczne "pisanie" skryptu na podstawie akcji, które zostały wykonane w systemie przez użytkownika. W ten sposób możliwe jest bardzo proste "nagranie" skryptu w celu późniejszego powtórzenia.

Faktem wartym zaznaczenia jest tutaj to, że skrypt wygenerowany poprzez Recorder wyprodukuje zazwyczaj kod znacznie większy niż konieczne. Wiąże się to głównie z tym, że na przykład gdy użytkownik chce kliknąć[[8]](#footnote-9) w coś, to przy wciśnięciu przycisku myszy Recorder nie ma pojęcia czy ma to być kliknięcie czy może wciśnięcie przycisku aby wykonać jakiegoś Drag&Drop'a. Jednak użytkownik, który ręcznie pisze taki skrypt zazwyczaj wie w jaki sposób chce użyć urządzenia.

* 1. Uruchomienie

Aby rozpocząć nagrywanie skryptu musimy uprzednio stworzyć nowy skrypt lub wybrać istniejący z listy. Jeśli zamierzamy wybrać istniejący skrypt wtedy Recorder "dogra" nowe komendy do skryptu. Po wybraniu miejsca na nasz skrypt klikamy w record.bmp aby rozpocząć nagrywanie (lub alternatywnie wciskamy przycisk F9).

* 1. Ustawienia

 Domyślnie przy nagrywaniu główne okno aplikacji zostanie schowane, a my wykonujemy akcje, które zostaną zapisane w skrypcie (domyślnie ruch myszy nie jest nagrywany, jedynie kliknięcia przycisków i klawiszy na klawiaturze). Istnieje możliwość zmodyfikowania, które urządzenia powinny być monitorowane oraz czy program powinien się "chować" po kliknięciu w przycisk nagrywania. Oprócz tego jest także możliwość ustalenia czy ruch myszy powinien być nagrywany. Domyślnie jest to wyłączone, gdyż ta opcja produkuje bardzo duży skrypt.

* 1. Pobieranie stanu ekranu

Gdy jesteśmy w trybie nagrywania skryptu możliwe jest pobranie fragmentu ekranu w celu jego znalezienie gdy odtwarzamy nagrywany skrypt. Aby przejść do wspomnianego trybu, podczas nagrywania naciskamy kombinację lewy ctrl + lewy shift + Z, po czym zostanie zmieniony kursor informujący o tym, że możemy już zaznaczać fragment ekranu. Robimy to zakreślając zwykły prostokąt jak gdybyśmy rysowali w programie MS Paint. Po zaznaczeniu fragmentu (zwolnieniu lewego przycisku myszy) nagrywanie wraca do normalnego trybu. Warto zaznaczyć, że jakiekolwiek działania myszką czy klawiaturą podczas trybu zaznaczania fragmentu ekranu nie są nagrywane.

Aby bardziej rozjaśnić sprawę omówmy to na jakimś przykładzie. Załóżmy, że chcemy zalogować się na jakiejś stronie. Czas wczytywania strony możemy być różny, zależnie od przeciążenia sieci czy innych czynników. Zatem musimy w jakiś sposób dowiedzieć się czy dana strona się już wczytała aby następnie wykonać kliknięcie w odpowiednim polu i wpisać dane.

Scenariusz taki możemy nagrać następująco:

1. Odpalamy Recorder'a i wchodzimy na stronę,
2. Wciskamy lewy ctrl + lewy shift + Z i zaznaczamy coś na stronie,
3. Klikamy w odpowiednie pola i wpisujemy dane.

Te *coś* w 3 punkcie powinno być fragmentem unikalnym dla całego ekranu (jeśli taki fragment występuje także gdzieś indziej, skrypt może się pomylić i pomyśleć, że nastał odpowiedni moment na wykonanie kolejnych komend). Ponadto fragment taki powinien być dosyć mały (aby można było go odnaleźć w rozsądnym czasie) ograniczając się jedynie do konkretnego obiektu z usuniętym nieużytecznym tłem. Np.:

|  |  |
| --- | --- |
| Gorzej | Lepiej |
| D:\DoForMe\DoForMe!\scripts\pic\pic0.bmp | D:\DoForMe\DoForMe!\scripts\pic\pic1.bmp |

W momencie odtwarzania takiego skryptu, zostanie on wstrzymany w przypadku gdy dany fragment nie zostanie znaleziony na ekranie. Aczkolwiek skrypt może sam się wznowić (o ile użytkownik osobiście nie przerwał skryptu), gdy dany fragment nagle pojawi się na ekranie.

* 1. Zatrzymanie

Gdy już wykonany wszystkie działania, które miały zostać nagrane wciskamy kombinację lewy ctrl + lewy shift + X aby zakończyć działanie Recorder'a. Po zakończeniu nagrywania zostaniemy o tym poinformowani stosownym komunikatem.

1. <http://en.wikipedia.org/wiki/Filename#Reserved_characters_and_words> [↑](#footnote-ref-2)
2. <http://en.wikipedia.org/wiki/Filename#Uniqueness> [↑](#footnote-ref-3)
3. dziwne, w sensie, że gdy wpiszemy sendText("{rshift+}") to po wykonaniu naszego skryptu shift będzie widziany w systemie jako wciśnięty i będziemy pisać dużymi literami mimo wyłączonego "caps lock'a" i nie wciśniętego fizycznie klawisza Shift (oczywiście można to szybko przywrócić do normalności klikając shift samemu) [↑](#footnote-ref-4)
4. dla każdej funkcji (w kontekście myszki) istnieje jej przeciążony odpowiednik z sufiksem At [↑](#footnote-ref-5)
5. istnieje funkcjonalność, która sprawdza czy dany obszar ekranu został już wyświetlony, zostanie to omówione podczas omawiania Recordera [↑](#footnote-ref-6)
6. aczkolwiek skrypty przy zapisywaniu są od razu zapisywane do pliku [↑](#footnote-ref-7)
7. oczywiście w kontekście systemu Windows [↑](#footnote-ref-8)
8. czyli wcisnąć i zwolnić - dla przypomnienia [↑](#footnote-ref-9)